Департамент образования Вологодской области
бюджетное профессиональное образовательное учреждение
Вологодской области
«ВОЛОГОДСКИЙ СТРОИТЕЛЬНЫЙ КОЛЛЕДЖ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к практическим работам
по МДК 02.01. Информационные технологии и платформы разработки информационных систем
ПМ.02. Участие в разработке информационных систем

09.02.04 Информационные системы (по отраслям)

2017

Рассмотрено на заседании предметно-цикловой комиссии общепрофессиональных, специальных дисциплин и дипломного проектирования по специальностям 08.02.01. Строительство и эксплуатация зданий и сооружений, 08.02.07. Монтаж и эксплуатация внутренних сантехнических устройств, кондиционирования воздуха и вентиляции, 43.02.08. Сервис домашнего и коммунального хозяйства

Данные методические указания предназначены для студентов специальности 09.02.04. Информационные системы (по отраслям) БПОУ ВО «Вологодский строительный колледж» при выполнении практических работ по МДК 02.01. Информационные технологии и платформы разработки информационных систем

Объем практических работ по МДК 02.01.составляет 90 часов.

Содержание

Практическая работа №36. Операторы языка программирования PHP. Операторы присваивания, арифметические операторы, операторы отношения, логические операторы, поразрядные операторы, строковые операторы. Управляющие конструкции языка PHP.
Практическая работа №37. Использование функций в PHP.
Практическая работа №38. Работа с массивами данных в PHP.
Практическая работа №39. Работа с формами. Способы работы форм. Методы GET и POST. Общие принципы обработки данных из форм. Доступные элементы форм и работа с ними.
Практическая работа №40. Работа с файлами и каталогами. Открытие, закрытие, чтение и запись, копирование, удаление и переименование файлов и каталогов.
Практическая работа №41. Работа с изображениями. Создание и вывод изображения. Рисование геометрических фигур. Работа с текстом. Примеры интерактивных изображений на странице.
Практическая работа №42.Проектирование базы данных и обеспечение прав доступа к ней.
Практическая работа №43. Взаимодействие PHP и XML. Установка расширения DOM XML. Обработка элементов XML документа с помощью функций PHP. Использование XML-базы данных в качестве альтернативы реляционной СУБД.
Практическая работа №44. Использование шаблонов в PHP. Понятие шаблона и его использование в языке программирования PHP. Классы шаблонов FastTemplate и Smarty.
Практическая работа №45. Работа с Cookies в PHP. Авторизация доступа с помощью сессий. Конфигурация PHP для работы с сессиями. Отслеживание сеанса в PHP.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

МДК 02.01. Информационные технологии и платформы разработки информационных систем входит в состав профессионального модуля ПМ.02. Участие в разработке информационных систем в составе профессионального учебного цикла программы подготовки специалистов среднего звена по специальности 09.02.04 Информационные системы (по отраслям).

В результате освоения ПМ.02. Участие в разработке информационных систем обучающийся должен **иметь практический опыт:**

- использования инструментальных средств обработки информации;
- участия в разработке технического задания;
- формирования отчетной документации по результатам работ;
- использования стандартов при оформлении программной документации;
- программирования в соответствии с требованиями технического задания;
- использования критериев оценки качества и надежности функционирования информационных систем;
- применения методики тестирования разрабатываемых приложений;
- управления процессом разработки приложений с использованием инструментальных средств;

уметь:

- осуществлять математическую и информационную постановку задач по обработке информации, использовать алгоритмы обработки информа-ции для различных приложений;
- уметь решать прикладные вопросы интеллектуальных систем с использованием, статических экспертных систем, экспертных систем реального времени;
- использовать языки структурного, объектно-ориентированного программирования и языка сценариев для создания независимых программ, разрабатывать графический интерфейс приложения;
- создавать проект по разработке приложения и формулировать его зада-чи, выполнять управление проектом с использованием инструментальных средств;

знать:

- основные виды и процедуры обработки информации, модели и методы решения задач обработки информации (генерация отчетов, поддержка принятия решений, анализ данных, искусственный интеллект, обработка изображений);
- сервисно - ориентированные архитектуры, CRM-системы, ERP-системы;
- объектно-ориентированное программирование;
- спецификации языка, создание графического пользовательского интер-фейса (GUI), файловый ввод-вывод, создание сетевого сервера и сетевого клиента;
- платформы для создания, исполнения и управления информационной системой;
- основные процессы управления проектом разработки.

В соответствии с учебным планом на изучение МДК 02.01. Информационные технологии и платформы разработки информационных систем отводится 306 часов, в том числе 90 часов – практические занятия.

Целью практических занятий является формирование практических умений, необходимых в последующей учебной и профессиональной деятельности.

Содержание практических занятий по МДК 02.01. Информационные технологии и платформы разработки информационных систем направлено на реализацию требований Федерального государственного образовательного стандарта по специальности СПО

09.02.04 Информационные системы (по отраслям). Практическое занятие включает следующие структурные элементы:

1) инструктаж, проводимый преподавателем,
2) самостоятельная деятельность обучающихся,
3) анализ и оценка выполненных работ.

Контроль и оценка результатов выполнения обучающимися работ, заданий на практических занятиях направлены на проверку освоения умений, практического опыта, развития общих и формирование профессиональных компетенций, определённых программой учебной дисциплины.

Оценки за выполнение заданий на практических занятиях выставляются по пятибалльной системе и учитываются как показатели текущей успеваемости обучающихся.

ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

МДК 02.01. Информационные технологии и платформы разработки информационных систем

№ п/п	Тема программы	Тема работы	Количество часов
1.	*Тема 1. Технологии обработки информации в информационных системах*	Моделирование логической задачи	2
2.		Моделирование экономической задачи Однофакторный дисперсионный анализ	2
3.		Моделирование экономической задачи. Двухфакторный дисперсионный анализ	2
4.		Моделирование оптимизационной задачи	2
5.		Решение задач линейного программирования	2
6.	*Тема 2. Задачи обработки информации в информационных системах*	Обобщение и анализ данных с использованием средств табличного процессора.	2
7.		Проектирование баз знаний с использованием средств табличного процессора.	2
8.		Сортировка базы данных	2
9.		Цифровая обработка изображений.	2
10.		Цифровая обработка видео.	2
11.	*Тема 3. Технологии интеллектуальных систем*	Решение прикладных задач с помощью статической экспертной системы.	2
12.		Решение прикладных задач с помощью экспертной системы реального времени.	2
13.	*Тема 4. Архитектура программного обеспечения информационных систем*	Работа со стандартами разработки программного обеспечения.	2
14.		Анализ отчетной документации различных этапов разработки информационных систем.	2
15.	*Тема 5. Представление данных в распределенных системах*	Создание простейших HTML-страниц в текстовом редакторе.	2
16.		Размещение на web-странице графических элементов. Организация гиперссылок.	2
17.		Размещение на web-странице таблиц.	2
18.		Создание web-страниц на основе фреймов.	2
19.		Создание HTML-форм.	2
20.		Проектирование дизайна web-узла.	2
21.		Наполнение web-узла контентом	2
22.		Разработка навигации web-узла.	2
23.		Создание макетов страниц web-узла.	2
24.		Разработка CSS-правил для управления оформлением web-узла.	2
25.		Разработка XML-документов.	2
26.	*Тема 6. Программы, выполняемые на стороне клиента*	Создание простейших сценариев.	2
27.		Программирование объекта Window.	2
28.		Методы объекта Window объекта Window.	2
29.		Программирование объекта Document	2
30.		Фреймы	2
31.		Встроенный класс Date.	2

32.		Создание интерактивных элементов web-страниц.	2
33.		Работа с формами.	2
34.		Работа с графическими изображениями.	2
35.	*Тема 7. Программы, выполняемые на стороне сервера*	Основы синтаксиса PHP. Оформление кода программы. Переменные, константы и типы данных языка программирования PHP.	2
36.		Операторы языка программирования PHP. Операторы присваивания, арифметические операторы, операторы отношения, логические операторы, поразрядные операторы, строковые операторы. Управляющие конструкции языка PHP.	2
37.		Использование функций в PHP.	2
38.		Работа с массивами данных в PHP.	2
39.		Работа с формами. Способы работы форм. Методы GET и POST. Общие принципы обработки данных из форм. Доступные элементы форм и работа с ними.	2
40.		Работа с файлами и каталогами. Открытие, закрытие, чтение и запись, копирование, удаление и переименование файлов и каталогов.	2
41.		Работа с изображениями. Создание и вывод изображения. Рисование геометрических фигур. Работа с текстом. Примеры интерактивных изображений на странице.	2
42.		Проектирование базы данных и обеспечение прав доступа к ней.	2
43.		Взаимодействие PHP и XML. Установка расширения DOM XML. Обработка элементов XML документа с помощью функций PHP. Использование XML-базы данных в качестве альтернативы реляционной СУБД.	2
44.		Использование шаблонов в PHP. Понятие шаблона и его использование в языке программирования PHP. Классы шаблонов FastTemplate и Smarty.	2
45.		Работа с Cookies в PHP. Авторизация доступа с помощью сессий. Конфигурация PHP для работы с сессиями. Отслеживание сеанса в PHP.	2

ПОРЯДОК ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ

Практическая работа №1
Моделирование логической задачи

Теоритический материал.

С помощью таблиц решаются задачи с четырьмя, пятью и более парами элементов, когда использование схем неудобно и не наглядно из-за чрезмерной громоздкости Рассмотрим задачу: В школе учатся четыре талантливых подростка: Иван, Петр, Алексей и Андрей. Один из них — будущий хоккеист, другой преуспел в футболе, третий — легкоатлет, четвертый подает надежды как баскетболист. О них известно следующее:

1. Иван и Алексей присутствовали в спортзале, когда там занимался легкоатлет.

2. Петр и хоккеист вместе были на тренировке баскетболиста.

3. Хоккеист раньше дружил с Андреем, а теперь неразлучен с Иваном

4. Иван незнаком с Алексеем, так как они учатся в разных классах и в разные смены. Кто чем увлекается?

Построим таблицу, в которой учтем все возможные варианты. Включим в нее столбцы с названиями: «Футболист», «Баскетболист», «Легкоатлет», «Хоккеист» и строки с именами мальчиков.

Из первого пункта следует, что ни Иван, ни Алексей не могут быть легкоатлетами. В таблице занесем в соответствующие клетки знак «минус».

Аналогично определяем, что:

Петр не баскетболист и не хоккеист (условие 2);

Андрей и Иван не хоккеисты (условие 3).

После анализа исходных условий таблица выглядит так:

	Футболист	Баскетболист	Легкоатлет	Хоккеист
Иван			-	-
Петр		-		-
Алексей			-	
Андрей				-

• По условию задачи каждый подросток обладает только одним талантом,следовательно, в каждой строчке и каждом столбце может быть только один «+».

• В графе «Хоккеист» оказалось три минуса, тогда хоккеистом должен быть Алексей, так как согласно условию хоккеист среди них есть. Поставим в этой клетке «+».

• Но раз Алексей хоккеист, он не может быть ни легкоатлетом, ни футболистом, ни баскетболистом, что и зафиксируем знаком «—» во всей «алексеевской» строчке.

	Футболист	Баскетболист	Легкоатлет	Хоккеист
Иван			-	-
Петр		-		-
Алексей	-	-	-	+

Андрей				-

• Сопоставим теперь второй и третий пункты условия задачи. Петр и Алексей вместе были на тренировке баскетболиста, но Иван не знает Алексея, значит, баскетболист — не Иван. Отметим этот факт минусом в соответствующей клетке.

• Теперь в столбике «Баскетболист» три минуса, поэтому баскетболистом является Андрей, ставим ему «+», а в оставшихся пустых клетках строки — минусы.

• Теперь определился легкоатлет — это Петр. Ставим минусы в его строке.

• Остается один Иван, и он, очевидно, футболист. Окончательный вид таблицы:

	Футболист	Баскетболист	Легкоатлет	Хоккеист
Иван	+	-	-	-
Петр	-	-	+	-
Алексей	-	-	-	+
Андрей	-	+	-	-

Таким образом, в результате составления логической модели в виде таблицы и анализа ее мы пришли к выводу, что Иван— футболист, Петр— легкоатлет, Алексей —хоккеист, а Андрей — баскетболист.

Задание №1: Решите задачу «В купе поезда»

В купе одного из вагонов поезда «Москва-Одесса» ехали москвич, петербуржец, туляк, киевлянин, харьковчанин и одессит. Их фамилии начинались с букв «А», «Б», «В», «Г», «Д», «Е». В дороге выяснилось, что:

1) А. и москвич — врачи;

2) Д. и петербуржец — учителя;

3) В. и туляк — инженеры;

4) Б. и Е. — участники Великой Отечественной войны, а туляк в армии совсем не служил;

5) харьковчанин старше А.;

6) одессит старше В.;

7) Б. и москвич сошли в Киеве;

8) В. и харьковчанин сошли в Виннице.

Определите профессию и место жительства каждого из пассажиров.

Задание №2:Решите задачу «Отъезд на работу»

У подъезда припарковались четыре машины: «Жигули», «Волга», «Москвич» и «Запорожец». Они красного, желтого, белого и зеленого цветов. По утрам на них уезжают художник, пекарь, учитель и инженер. Происходит это в 8, 9, 10 и 12 часов.

Определите, кому какая машина принадлежит, какого она цвета и когда отъезжает, если известно, что:

1) у учителя не «Москвич»; .

2) «Запорожец» не белый, а «Волга» не желтая;

3) инженер уезжает сразу после «Запорожца»;

4) пекарь уезжает не в 9 и не в 12;

5) раньше всех уезжает учитель;

6) художник уезжает позже красного «Москвича».

Практическая работа №2

Моделирование экономической задачи. Однофакторный дисперсионный анализ

Дисперсионный анализ предназначен для исследования задачи о действии на измеряемую случайную величину (отклик) одного или нескольких независимых факторов.

В MS Excel для проведения однофакторного дисперсионного анализа используется процедура **Однофакторный дисперсионный анализ.**

Для проведения дисперсионного анализа необходимо:

1. ввести данные в таблицу, так чтобы в каждом столбце оказались данные, соответствующие одному значению исследуемого фактора, а столбцы располагались в порядке возрастания (убывания) величины исследуемого фактора;

2. выполнить команду **Сервис** > **Анализ данных**;

3. в появившемся диалоговом окне **Анализ данных** в списке **Инструменты анализа** выбрать процедуру **Однофакторный дисперсионный анализ**, затем нажать кнопку **ОК**;

4. в появившемся диалоговом окне задать **Входной интервал**, то есть ввести ссылку на диапазон анализируемых данных, содержащий все столбцы данных.

5. в разделе **Группировка** переключатель установить в положение *по столбцам*;

6. указать **выходной интервал**, то есть ввести ссылку на ячейку, с которой будут показаны результаты анализа. Размер выходного диапазона будет определен автоматически, и на экран будет выведено сообщение в случае возможного наложения выходного диапазона на исходные данные. Нажать кнопку **ОК.**

Выходной диапазон будет включать в себя результаты дисперсионного анализа: средние, дисперсии, критерий Фишера и другие показатели. Влияние исследуемого фактора определяется по величине значимости критерия Фишера, которая находится в таблице **Дисперсионный анализ** на пересечении строки **Между группами** и столбца **Р-Значение**. В случаях, когда Р-Значение < 0,05, критерий Фишера значим и влияние исследуемого фактора можно считать доказанным.

Кроме рассмотренной процедуры однофакторного дисперсионного анализа, для проведения двухфакторного дисперсионного анализа в пакете анализа реализованы процедуры **Двухфакторный дисперсионный анализ с повторениями** и **Двухфакторный дисперсионный анализ без повторений.**

Пример 1. Необходимо выявить, влияет ли расстояние от центра города на степень заполняемости гостиниц. Пусть введены 3 уровня расстояний от центра города: 1) до 3 км, 2) от 3 до 5 км и 3) свыше 5 км. Данные заполняемости представлены в таблице 1.

Решение

Таблица 1.

Расстояние	Заполняемость
До 3 км	92 98 89 97 90 94
От 3 до 5 км	90 86 84 91 83 82
Свыше 5 км	87 79 74 85 73 77

1. Исследуемые данные введите в рабочую таблицу Excel по столбцам: в столбец
A — заполняемость гостиниц в центре города, в столбец В — гостиниц, находящихся на расстоянии от 3 до 5 км и т. д. (диапазон A1:C6).

2. Выполните команду **Сервис** > **Анализ данных**. В появившемся диалоговом окне **Анализ данных** в списке **Инструменты анализа** щелчком мыши выберите процедуру **Однофакторный дисперсионный анализ**. Нажмите кнопку **ОК.**

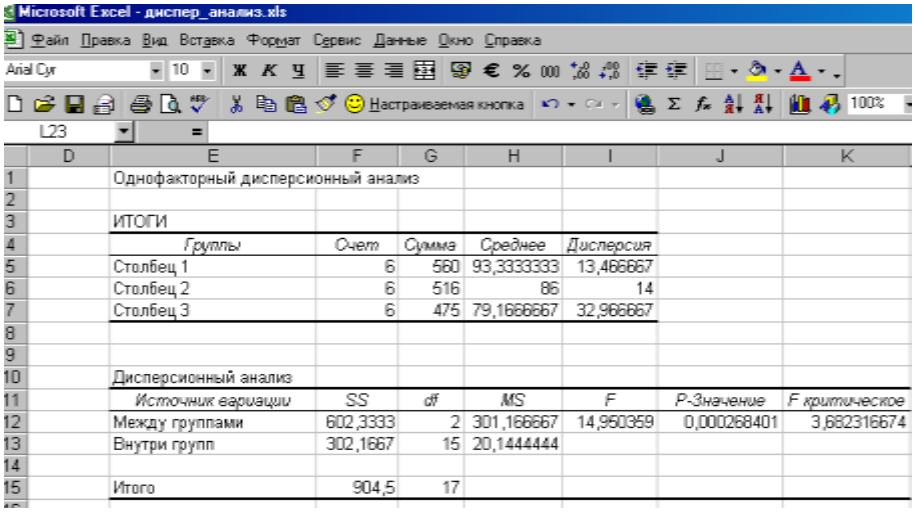
3. В появившемся диалоговом окне Однофакторный дисперсионный анализ в поле **Входной интервал** задайте *A1:C6.*

4. В разделе **Группировка** переключатель установите в положение по столбцам.

5. Укажите выходной диапазон *E1* и нажмите **Ок.**

В результате будет получена следующая таблица см. рис. 1. В таблице **Дисперсионный анализ** на пересечении строки **Между группами** и столбца Р-значение находится величина 0,0002684 <0,05, следовательно, критерий Фишера значим и влияние

фактора расстояния от центра города на эффективность заполнения гостиниц доказана статистически.



Microsoft Excel - диспер_анализ.xls

Однофакторный дисперсионный анализ

ИТОГИ

Группы	Счет	Сумма	Среднее	Дисперсия
Столбец 1	6	560	93,3333333	13,466667
Столбец 2	6	516	86	14
Столбец 3	6	475	79,1666667	32,966667

Дисперсионный анализ

Источник вариации	SS	df	MS	F	P-Значение	F критическое
Между группами	602,3333	2	301,166667	14,950359	0,000268401	3,682316674
Внутри групп	302,1667	15	20,1444444			
Итого	904,5	17				

Рис. 1. Результаты примера 1

Задание для самостоятельной работы

1. Определите влияет ли фактор образования на уровень зарплаты сотрудников фирмы на основании следующих данных (см. табл. 2).

Таблица 2.

Образование	Зарплата сотрудников					
Высшее	3200	3000	2600	2000	1900	1900
Среднее спец.	2600	2000	2000	1900	1800	1700
среднее	2000	2000	1900	1800	1700	1700

2. Исследователь сравнивает эффективность четырех разных методик обучения производственным навыкам. Для этой цели из всех выпускников ПТУ выбраны четыре группы учащихся, обучавшиеся, соответственно четырьмя разными методами. Эффективность методик оценивалась по сумме обработанных учащимися деталей в течение одного дня (см. табл. 3). Проверить гипотезу об отсутствии влияния регулируемого фактора (методик обучения) на продуктивность деятельности ученика (Ермолаев О.Ю. Математическая статистика для психологов: Учебник, М., 2003. – стр. 192).

Таблица 3.

№ учащихся	1 группа	2 группа	3 группа	4 группа
1	60	75	60	95
2	80	66	80	85
3	75	85	65	100
4	80	80	60	80
5	85	70	86	
6	70	80	75	
7		90		

3. Проведите дисперсионный анализ для примера 1, задания 1 и задания 2 в статистическом пакете Stadia (см. лаб. 5). Для этого выбираем процедуру **B=Однофакторный** в окне **Статистические методы – Дисперсионный анализ**. При запросе метода выбираем **1=параметрический.** Совпадают ли полученные значения.

Практическая работа №3

Моделирование экономической задачи. Двухфакторный дисперсионный анализ

Цель работы: сформировать умения по проведению однофакторного и двухфакторного дисперсионного анализа.

Задания:

1. Изучить особенности использования инструментов **MS Excel** для проведения однофакторного и двухфакторного дисперсионного анализа.
2. В среде табличного процессора **MS Excel** решить задачи демонстрационных примеров.

Теоретические сведения:

- **Дисперсионный анализ**

Метод дисперсионного анализа служит для установления влияния отдельных факторов на изменчивость какого-либо признака, значения которого могут быть получены опытным путем в виде случайной величины **Y**. При этом величину **Y** называют результативным признаком, а конкретную реализацию фактора **A** - уровнем (группой) фактора **A** или способом обработки.

В зависимости от числа оказывающих влияние факторов различают однофакторный и многофакторный (двухфакторный и т. д.) дисперсионный анализ.

Идея метода: пусть $a_1, a_2, \ldots, a_m$ - математическое ожидание результативного признака соответственно при уровне *A(1), A(2), …, A(m)*. Если при изменении уровня фактора групповые математические ожидания не изменяются, т. е. $a_1 = a_2 = \ldots = a_m$, то считаем, что результативный признак не зависит от фактора **A**, в противном случае такая зависимость имеется. Но поскольку числовые значения математических ожиданий неизвестны, возникает задача проверки гипотезы $H_0 : a_1 = a_2 = \ldots = a_m$.

Проверить гипотезу о равенстве групповых математических ожиданий можно, соблюдая следующие требования при каждом уровне фактора:

1) наблюдения независимы и проводятся в одинаковых условиях;
2) результативный признак имеет нормальный закон распределения с постоянной для различных уровней генеральной дисперсией σ^2.

При этом возникает вопрос, как установить, одинаковы генеральные дисперсии результативного признака при различных уровнях фактора или нет? Не зная числовых значений этих дисперсий, нельзя однозначно ответить на этот вопрос, можно лишь проверить гипотезу: $H_0 : \sigma_1^2 = \sigma_2^2 = \ldots = \sigma_m^2$.

Для проверки гипотезы о равенстве дисперсий трех (и более) нормальных распределений применяется критерий Бартлетта.

Если в процессе анализа выявлено влияние фактора **A** на результативный признак **Y**, то можно измерить степень данного влияния с помощью выборочного коэффициента детерминации: $коэффициент\,детерминации = \frac{дисперсия\ групповых\ средних}{общая\ выборочная\ дисперсия}$, который показывает, какая доля выборочной дисперсии объясняется зависимостью результативного признака **Y** от влияющего фактора **A**.

- **Однофакторный дисперсионный анализ**

Однофакторный дисперсионный анализ позволяет по выборочным данным выяснить, влияет ли контролируемый фактор на результативный признак, и при наличии такого влияния оценить его степень.

Режим работы **Однофакторный дисперсионный анализ** из **Пакета анализа** служит для выяснения факта влияния контролируемого фактора **A** на результативный признак **Y** на основе выборочных данных.

В диалоговом окне данного режима задаются следующие параметры:

1. Входной интервал.
2. Группирование.
3. Метки в первой строке, Метки в первом столбце.
4. Альфа - вводится уровень значимости *a*, равный вероятности возникновения ошибки первого рода (отвержение нулевой гипотезы).
5. Выходной интервал: Новый рабочий лист, Новая рабочая книга.

- **Двухфакторный дисперсионный анализ**

Процедура двухфакторного дисперсионного анализа позволяет решать задачи о действии на результативный признак **Y** двух факторов - **A** и **B**. Такие задачи характерны как для промышленных и технологических экспериментов, так и для гуманитарных исследований. Типичный пример - выяснение зависимости качества пряжи от типа станка и вида сырья, из которой она изготавливается.

Логика однофакторного и двухфакторного дисперсионного анализа во многом схожа. Аналогичным образом выдвигаются гипотезы: $H_A : a_1 = a_2 = ... = a_{m_A}$, $H_B : b_1 = b_2 = ... = b_{m_B}$.

Проверка выдвинутых гипотез осуществляется так же, как и при однофакторном дисперсионном анализе, и состоит в нахождении правосторонних критических интервалов $\left(F^{кр}_{пр,\alpha}, +\infty\right)$ с последующим контролем попадания (или непопадания) в данный интервал расчетных значений F_p^A (или F_p^B). Если расчетное значение попадает в критический интервал, то гипотеза H_A (или H_B) отвергается, т.е. считается, что фактор **A** (или **B**) влияет на результативный признак **Y.**

Двухфакторный дисперсионный анализ может иметь две разновидности: без повторений и с повторениями. В первом случае каждому уровню факторов соответствует только одна выборка данных, во втором - определенным уровням факторов может соответствовать более одной выборки данных.

Режимы работы **Двухфакторный дисперсионный анализ без повторений** и **Двухфакторный дисперсионный анализ с повторениями** служат для выяснения на основе выборочных данных факта влияния контролируемых факторов **A** и **B** на результативный признак **Y**. При этом в режиме **Двухфакторный дисперсионный анализ без повторений** каждому уровню факторов **A** и **B** соответствует только одна выборка данных, а в режиме **Двухфакторный дисперсионный анализ с повторениями** каждому уровню одного из факторов **A** (или **B**) соответствует более одной выборки данных. В последнем случае число выборок для каждого уровня должно быть одинаковым.

В диалоговых окнах данных режимов задаются те же параметры, что и в диалоговом окне **Однофакторный дисперсионный анализ**, только добавлено поле **Число строк для выборки** в это поле вводится число выборок, приходящихся на каждый уровень одного из факторов; каждый уровень фактора должен содержать одно и тоже количество выборок (строк таблицы).

Демонстрационные примеры:

Пример 1. Выборочные данные об объеме работ, выполненных на стройке (за смену) четырьмя бригадами, приведены в таблице, сформированной на рабочем листе **Microsoft Excel**.

	A	B	C	D	E
1	Объем выполненной работы				
2	Номер смены	Бригада 1	Бригада 2	Бригада 3	Бригада 4
3	1	140	150	148	150
4	2	144	149	149	155
5	3	142	152	146	154
6	4	145	150	147	152
7					

При уровне значимости $a = 0,05$ требуется выяснить, зависит ли объем выполненных работ от работающей бригады.

Прежде чем проводить анализ данных в сгенерированных таблицах, покажем, как с помощью критерия Бартлетта проверить гипотезу о равенстве генеральных дисперсий $H_0 : \sigma_1^2 = \sigma_2^2 = \sigma_3^2 = \sigma_4^2$.

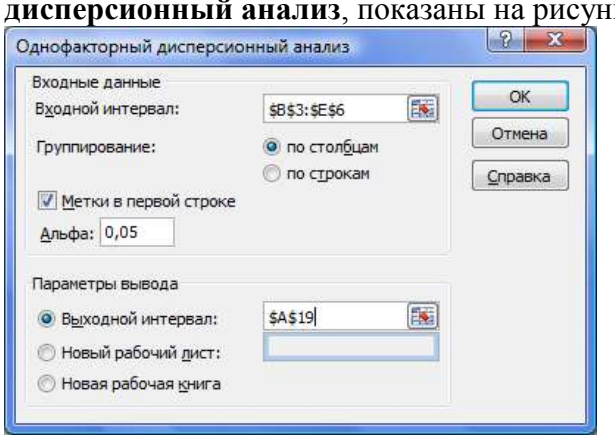
Рассчитайте для примера 1 параметры данной гипотезы, и представьте их в таблице. Ячейки таблицы содержат следующие формулы:

- в массиве **B10:E10** определяются объемы выборок n_i (например, ячейка **C10** содержит формулу **=СЧЁТ(B3:B6)**);
- в массиве **B11:E11** вычисляются несмещенные оценки s_i^2 групповых дисперсий (например, ячейка **B11** содержит формулу **=ДИСП(B3:B6)**);
- ячейка **B12** содержит формулу **{=СУММПРОИЗВ(B10:E10-1;B11:E11)/СУММ(B10:E10-1)}** рассчитывается объединенная оценка s^2;
- ячейка **B13** содержит формулу **{=1/(1+1/(3*(4-1))*(СУММ(1/(B10:E10-1))-1/СУММ(B10:E10-1)))}** вычисляется значение коэффициента q;
- ячейка **B14** содержит формулу **{=B13*СУММПРОИЗВ(B10:E10-1;LN(B12/B11:E11))}** - рассчитывается значение критерия Бартлетта w_p;
- ячейка **B16** содержит формулу **=ХИ2ОБР(0,05;3)** – определяется значение правосторонней критической точки $w_{np,\alpha}^{кр}$.

	A	B	C	D	E
1	Объем выполненной работы				
2	Номер смены	Бригада 1	Бригада 2	Бригада 3	Бригада 4
3	1	140	150	148	150
4	2	144	149	149	155
5	3	142	152	146	154
6	4	145	150	147	152
7					
8					
9					
10	Число наблюдений	4	4	4	4
11	Оценки s_i^2	4,92	1,58	1,67	4,92
12	Оценки S^2	3,27	3,27	3,27	3,27
13	q	0,88	0,88	0,88	0,88
14	w_p	1,53			
15	$W_{пр}^{кр}$	7,81			
16					

Так как $w_p = 1{,}53$ не попадает в критическую область $(7{,}81, +\infty)$, то гипотеза $H_0 : \sigma_1^2 = \sigma_2^2 = \sigma_3^2 = \sigma_4^2$ принимается и можно приступить к проверке гипотезы $H_0 : a_1 = a_2 = \ldots = a_m$.

Для решения задачи используем режим работы **Однофакторный дисперсионный анализ**. Значения параметров, установленных в диалоговом окне **Однофакторный дисперсионный анализ**, показаны на рисунке.



Показатели, рассчитанные в ходе проверки данной гипотезы, представлены в таблице:

Однофакторный дисперсионный анализ

ИТОГИ

Группы	*Счет*	*Сумма*	*Среднее*	*Дисперсия*
Бригада 1	4	571	142,75	4,916666667
Бригада 2	4	601	150,25	1,583333333
Бригада 3	4	590	147,5	1,666666667
Бригада 4	4	611	152,75	4,916666667

Дисперсионный анализ

сточник вариаци	*SS*	*df*	*MS*	*F*	*P-Значение*	*F критическое*
Между группами	220,1875	3	73,3958333	22,43949045	3,28029E-05	3,490294821
Внутри групп	39,25	12	3,27083333			
Итого	259,4375	15				

Данная таблица называется **таблицей однофакторного дисперсионного анализа.** Расчетное значение F-критерия $F_p = 22{,}44$, а критическая область образуется правосторонним интервалом $(3{,}49, +\infty)$. Так как F_p попадает в критическую область, то гипотезу H_0 о равенстве групповых математических ожиданий отвергаем, те, считаем, что объем ежедневной выборки зависит от работающей бригады.

- В ячейке **B31** (показатель *SS* между группами) рассчитывается взвешенная сумма квадратов отклонений групповых средних от общей выборочной средней (дисперсия групповых средних).
- В ячейке **B32** (показатель *SS* внутри групп) вычисляется остаточная сумма квадратов отклонений наблюдаемых значений уровня от своей выборочной средней.
- В ячейке **B33** (показатель *SS* итого) рассчитывается общая сумма квадратов отклонений наблюдаемых значений от общей выборочной средней (общая выборочная дисперсия).
- В ячейках **C31:C33** (показатель *df*) определяются степени свобод.
- В ячейках **D31:D32** (показатель *MS*) вычисляются несмещенные оценки.

- В ячейке **E31** (показатель *F*) вычисляется расчетное значение критерия F_p
- В ячейке **F31** (показатель *P-значение*) определяется *P-значение*, соответствующее расчетному значению критерия F_p.
- В ячейке **G31** (показатель *F критическое*) рассчитывается значение правосторонней критической точки $F^{кр}_{пр,\alpha}$.

Задание: Рассчитайте коэффициент детерминации для **примера 1** и сделайте соответствующий вывод.

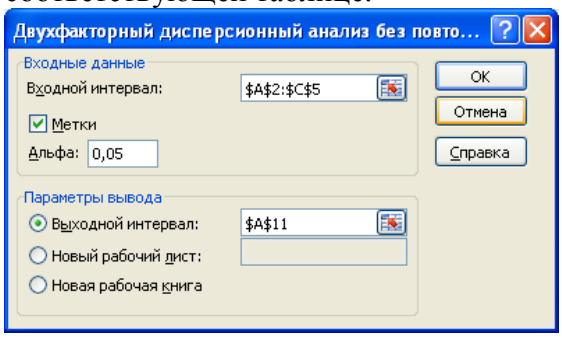
В соответствующей ячейке введите формулу: **=B31/B34** . Полученный коэффициент детерминации **0,85** показывает, что **85%** общей выборочной вариации ежедневного объема выработки связано с работающей бригадой.

Пример 2. Выборочные данные о разрывной нагрузке пряжи, изготовленной на разных станках и из отличающегося некоторым образом друг от друга сырья, приведены в таблице, сформированной на рабочем листе Microsoft Excel.

	A	B	C
1	Тип станка	Вид сырья	
2		Шелк натуральный	Шелк искусственный
3	JANOME	10	50
4	HUSQVARNA	20	60
5	SINGER	30	100

Требуется при уровне значимости $a = 0,05$ выяснить, влияют ли на качество пряжи, измеряемое величиной разрывной нагрузки, тип станка и вид сырья, из которого пряжа производится.

Для решения задачи используем режим работы **Двухфакторный дисперсионный анализ без повторений**. Значения параметров, установленных в одноименном диалоговом окне, представлены на рисунке, а рассчитанные в данном режиме показатели – в соответствующей таблице.



Двухфакторный дисперсионный анализ без повторений					
ИТОГИ	*Счет*	*Сумма*	*Среднее*	*Дисперсия*	
JANOME	2	60	30	800	
HUSQVARNA	2	80	40	800	
SINGER	2	130	65	2450	
Шелк натураль	3	60	20	100	
Шелк искусстве	3	210	70	700	
Дисперсионный анализ					
точник вариац	*SS*	*df*	*MS*	*F*	*P-Значение*	*F критическое*
Строки	1300	2	650	4,333333	0,1875	19
Столбцы	3750	1	3750	25	0,037749551	18,51282051
Погрешность	300	2	150			
Итого	5350	5				

Данная таблица является таблицей двухфакторного дисперсионного анализа без повторений. Расчетное значение *F-критерия* фактора A (тип станка) $F_p^A = 4{,}33$, а критическая область образуется правосторонним интервалом $(19,+\infty)$. Так как F_p^A не попадает в критическую область, то гипотезу H_A принимаем, т.е. считаем, что влияние типа станков на качество пряжи не подтвердилось.

Расчетное значение *F-критерия* фактора B (вид сырья) $F_p^B = 25$, а критическая область образуется правосторонним интервалом $(18{,}51,+\infty)$. Так как F_p^B попадает в критическую область, то гипотезу H_B отвергаем, т.е. считаем, что вид сырья влияет на качество пряжи.

Выборочный коэффициент детерминации равен 0,7, и показывает, что 70 % общей выборочной вариации качества пряжи связано с влиянием на нее вида сырья.

Пример 3. Выборочные данные об урожайности пшеницы, выращенной на участках, на которые вносились различные виды удобрений и которые подвергались различной химической обработке, приведены в таблице, сформированной на рабочем листе Microsoft Excel.

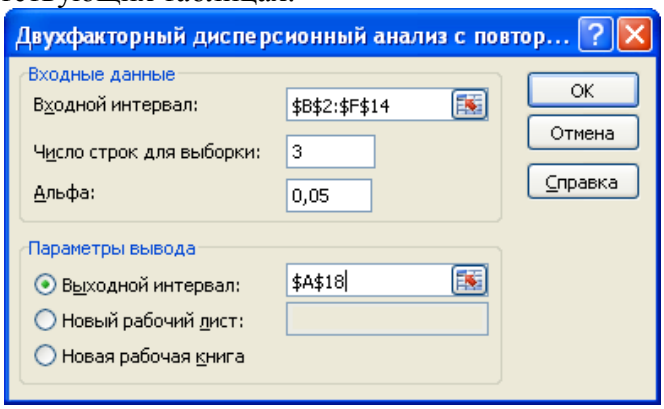
	A	B	C	D	E	F
1	**Номер участка**	**Вид удобрения**	**Способ химической обработки**			
2			**Способ 1**	**Способ 2**	**Способ 3**	**Способ 4**
3	Участок 1	Удобрение 1	21,4	20,9	19,6	17,6
4	Участок 2		21,2	20,3	18,8	16,6
5	Участок 3		20,1	19,8	16,4	17,5
6	Участок 1	Удобрение 2	12	13,6	13	13,3
7	Участок 2		14,2	13,3	13,7	14
8	Участок 3		12,1	11,6	12	13,9
9	Участок 1	Удобрение 3	13,5	14	12,9	12,4
10	Участок 2		11,9	15,6	12,9	13,7
11	Участок 3		13,4	13,8	12,1	13
12	Участок 1	Удобрение 4	12,8	14,1	14,2	12
13	Участок 2		13,8	13,2	13,6	14,6
14	Участок 3		13,7	15,3	13,3	14
15						

Требуется при уровне значимости $a = 0,05$ выяснить, влияют ли на урожайность пшеницы вид удобрения и способ химической обработки почвы.

Рассматриваемый в задаче эксперимент представляет собой факторный эксперимент типа **4x4**, при котором четыре вида удобрений (фактор **A**) пересекаются с использованием четырех способов химической обработки почвы (фактор B). Таким образом, в плане эксперимента имеется **16 условий.**

Но в отличие от ранее рассмотренной ранее задачи здесь каждому условию соответствует не одно, а три значения (3 участка земли, засеянных пшеницей).

Для решения задачи используем режим работы **Двухфакторный дисперсионный анализ с повторениями**. Значения параметров, установленных в одноименном диалоговом окне, представлены на рисунке, а рассчитанные в данном режиме показатели в соответствующих таблицах.



16	Двухфакторный дисперсионный анализ с повторениями					
17						
18	ИТОГИ	Способ 1	Способ 2	Способ 3	Способ 4	Итого
19	*Удобрение 1*					
20	Счет	3	3	3	3	12
21	Сумма	62,7	61	54,8	51,7	230,2
22	Среднее	20,9	20,333333	18,266667	17,233333	19,18333333
23	Дисперсия	0,49	0,3033333	2,7733333	0,3033333	3,134242424
24						
25	*Удобрение 2*					
26	Счет	3	3	3	3	12
27	Сумма	38,3	38,5	38,7	41,2	156,7
28	Среднее	12,76666667	12,833333	12,9	13,733333	13,05833333
29	Дисперсия	1,543333333	1,1633333	0,73	0,1433333	0,819015152
30						
31	*Удобрение 3*					
32	Счет	3	3	3	3	12
33	Сумма	38,8	43,4	37,9	39,1	159,2
34	Среднее	12,93333333	14,466667	12,633333	13,033333	13,26666667
35	Дисперсия	0,803333333	0,9733333	0,2133333	0,4233333	0,986060606
36						
37	*Удобрение 4*					
38	Счет	3	3	3	3	12
39	Сумма	40,3	42,6	41,1	40,6	164,6
40	Среднее	13,43333333	14,2	13,7	13,533333	13,71666667
41	Дисперсия	0,303333333	1,11	0,21	1,8533333	0,726969697
42						
43	*Итого*					
44	Счет	12	12	12	12	
45	Сумма	180,1	185,5	172,5	172,6	
46	Среднее	15,00833333	15,458333	14,375	14,383333	
47	Дисперсия	13,25901515	9,7062879	6,3893182	3,519697	
48						

50	Дисперсионный анализ						
51	*Источник вариации*	*SS*	*df*	*MS*	*F*	*P-Значение*	*F критическое*
52	Выборка	309,2589583	3	103,08632	123,64176	1,11281E-17	2,901119588
53	Столбцы	9,970625	3	3,3235417	3,9862569	0,016084218	2,901119588
54	Взаимодействие	25,67854167	9	2,8531713	3,4220945	0,004729788	2,188765768
55	Внутри	26,68	32	0,83375			
56							
57	Итого	371,588125	47				
58							

Данная таблица является таблицей двухфакторного дисперсионного анализа с повторениями. Расчетное значение *F-критерия* фактора **A** (вид удобрения) $F_p^A = 123{,}64$, а критическая область образуется правосторонним интервалом $(2{,}90, +\infty)$. Так как F_p^A попадает в критическую область, то гипотезу H_A отвергаем, т. е. считаем, что вид удобрения влияет на урожайность пшеницы.

Выборочный коэффициент детерминации для фактора **A** равен **0,83** и показывает, что **83%** общей выборочной вариации урожайности пшеницы связано с влиянием вида удобрения.

Расчетное значения F -критерия фактора B (способ химической обработки) $F_p^B = 3{,}99$, а критическая область образуется правосторонним интервалом $(2{,}90,+\infty)$. Так как F_p^B попадает в критическую область, то гипотезу H_B отвергаем, т. е. считаем, что способ химической обработки почвы также влияет на урожайность пшеницы.

Выборочный коэффициент детерминации для фактора **B** равен **0,03** и показывает, что только около **3 %** общей выборочной вариации урожайности пшеницы связано с влиянием способа химической обработки почвы.

Значимость фактора взаимодействия $F_p^{AB} = 3{,}42$ и попадает в критический интервал $(2{,}19,+\infty)$. Это указывает на то, что эффективность различных видов удобрения варьируется при различных способах химической обработки почвы.

Внеаудиторная самостоятельная работа

Составить конспект по основным понятиям и командам, приведенным в работе.

Практическая работа №4
Моделирование оптимизационной задачи

Теоретический материал

В сфере управления сложными системами (например, в экономике) применяется оптимизационное моделирование, в процессе которого осуществляется поиск наиболее оптимального пути развития системы.

Критерием оптимальности могут быть различные параметры, например, в экономике можно стремиться к максимальному количеству выпускаемой продукции, а можно - к ее низкой себестоимости. Оптимальное развитие соответствует экстремальному (максимальному или минимальному) значению выбранного целевого параметра.

Развитие сложных систем зависит от множества факторов (параметров), следовательно, значение целевого параметра зависит от множества параметров. Выражением такой зависимости является целевая функция

K=F(x1, x2, ..., xn)

где К - значение целевого параметра; x1, x2, ..., xn - параметры, влияющие на развитие системы.

Цель исследования состоит в нахождении экстремума этой функции и определения значений параметров, при которых этот экстремум достигается. Если целевая функция нелинейная, то она имеет экстремумы, которые находятся определенными методами.

Однако часто целевая функция линейна и, соответственно, экстремумов не имеет. Задача поиска оптимального режима при линейной зависимости приобретает смысл только при наличии определенных ограничений на параметры. Если ограничения на параметры (системы неравенств) также имеют линейный характер, то такие задачи являются задачами *линейного программирования*.

Решение задач оптимизации состоит в поиске оптимального плана с использованием математических моделей и вычислительных методов, которые реализуются с помощью компьютеров и специальных программ-оптимизаторов.

Процедуру **поиска решения** можно использовать для определения значения влияющей ячейки, которое соответствует экстремуму зависимой ячейки - например можно изменить объем планируемого бюджета рекламы и увидеть, как это повлияет на проектируемую сумму расходов.

Для решения общей **оптимизационной задачи** в **Excel** с использованием настройки **Поиск решения** следует выполнить следующие действия:

1. Ввести формулу целевой функции, ссылающуюся на влияющие ячейки;
2. Ввести формулы ограниченй **оптимизационной задачи;**
3. Выбрать в **Excel** пункт меню *Поиск решения*;
4. В окне *Поиск решения* выбрать целевую ячейку, изменяемые ячейки и добавить ограничения;
5. Нажать кнопку *Выполнить*, после чего будет получено решение **оптимизационной задачи.**

Рассмотрим пример решения задачи линейного программирования.
Формулировка: Предприятие имеет запасы 4-х видов ресурсов (мука, жиры, сахар, финансы), с которых производится 2 виды продуктов (хлеб и батон). Известны:

Ресурсы	Хлеб	Батон	Запасы
Мука	0,6	0,5	120
Жиры	0,05	0,08	70
Сахар	0,2	0,6	65
Финансы	0,2	0,24	50
цена	0,99	1,21	

- нормы расходов ресурсов на производство единицы продунции;
- запасы ресурсов;
- цены продуктов;
- спрос на хлеб.

Найти оптимальный план производства, при котором доход от реализации произведенной продукции должен быть максимальным.

Экономико-математическая модель.

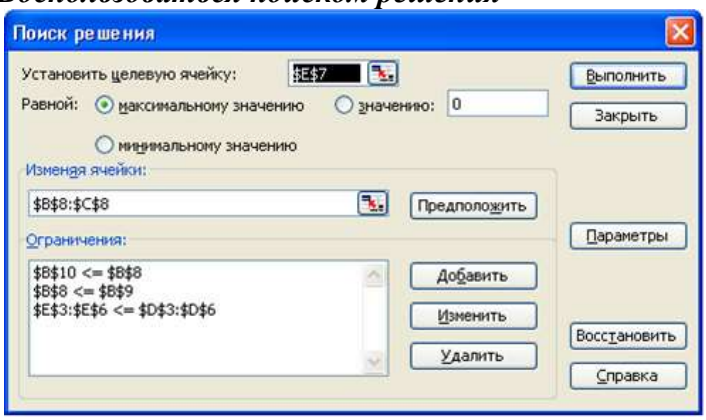
Найти план (количество хлеба и батонов) такой, чтобы
*Доход =0,99*Хлеб+1,21*Батон - мах*
При ограничениях:
*0,6*Хлеб+0,5*Батон<=120*
*0,05*Хлеб+0,08*Батон<=70*
*0,2*Хлеб+0,6*Батон<=65*
*0,2*Хлеб+0,24*Батон<=50*
120<=Хлеб<=150, а также Батон>=0

Реализация в Excel

Использованные ресурсы для ячейки Е3 находятся по формуле *=СУММПРОИЗВ(B3:C3;$B$8:$C$8)*

	A	B	C	D	E	F
1	*Оптимизационный план производства*					
2	**Ресурсы**	**Хлеб**	**Батон**	**Запасы**	**Использовано**	**Остаток**
3	Мука	0,6	0,5	120	0	120
4	Жиры	0,05	0,08	70	0	70
5	Сахар	0,2	0,6	65	0	65
6	Финансы	0,2	0,24	50	0	50
7	Цена	0,99	1,21	***Доход***	0	
8	План					
9	Спрос (верх.)	150				
10	Спрос (ниж.)	120				

Воспользоваться поиском решения



Полученный результат

	A	B	C	D	E	F
1	*Оптимизационный план производства*					
2	**Ресурсы**	**Хлеб**	**Батон**	**Запасы**	**Использовано**	**Остаток**
3	Мука	0,6	0,5	120	119,2	0,8
4	Жиры	0,05	0,08	70	12,2	57,8
5	Сахар	0,2	0,6	65	65,0	0,0
6	Финансы	0,2	0,24	50	44,0	6,0
7	Цена	0,99	1,21	***Доход***	219,1	
8	План	150	58,3			
9	Спрос (верх.)	150				
10	Спрос (ниж.)	120				

Контрольное задание:

1. Проверить задачу, представленную выше. Написать вывод по модели. Сколько булок хлеба и батонов нужно производить для получения максимальной прибыли.
2. Решить задачу в Excel.

	A	B	C	D	E
1	Сколько производить и сколько купить				
2		Прибор 1	Прибор 2	Прибор 3	
3	Заказ	3010	2000	1003	Запасы
4	Материал 1	2	1	3	10000
5	Материал 2	1	2	1,5	5000
6	Затраты на производство	47	80	125	
7	Затраты на покупку	63	95	140	

Практическая работа №5

Решение задач линейного программирования

Задание. Решить оптимизационную задачу в Excel. Найти оптимальный план производства, при котором доход от реализации произведенной продукции должен быть максимальным.

Задача. Завод получил заказ на выпуск приборов 3 типов, для их производства есть запасы материалов 2 типов. Если этих запасов будет мало, завод будет должен купить часть приборов на другом заводе. Нужно определить план исполнения заказа, при котором затраты будут минимальные. В таблице наведены данные относительно заказа, запасов, норм затрат материалов и затрат на производство и покупку.

Экономико-математическая модель.

1. *Для решения этой задачи нам нужно определить 6 неизвестных: П1, П2, П3 - количество приборов трех типов, которые нужно произвести на заводе; К1, К2, К3 - количество приборов трех типов, которые нужно купить.*
2. *Общие затраты = 47П1+80П2+125П3+63К1+95К2+140К3 - min.*
3. *При ограничениях: 2П1+1П2+3П3<=10000; 1П1+2П2+1,5П3<=5000;*

П1+К1=3010; П2+К2=2000; П3+К3=1000

и придельных ограничениях: П1, П2, П3, К1, К2, К3>=0.

3. Оформить отчет.

Практическая работа №6

Обобщение и анализ данных с использованием средств табличного процессора

Методические указания:

1. Общие сведения о прогнозировании

Прогноз, в переводе с греческого, означает "вперед, узнавание". В применении к экономике можно сказать, что *прогноз - это научно обоснованное суждение о возможных состояниях экономического объекта в будущем, альтернативных путях и сроках их реализации.*

Основными особенностями прогноза являются:

- прогноз как итог выводов, эмпирических данных и обоснованных предположений, представляет собой аргументированное заключение о направлении развития объекта или процесса в будущем;
- возникновение будущего как следствия прошедших или настоящих событий имеет элемент случайности, поэтому прогноз должен иметь оценку степени вероятности наступления события;
- для составления прогноза должны использоваться научные исследования количественного и качественного характера;
- прогноз является ориентиром для планирования;
- при разработке прогноза явления действительности абстрагируются.

Экономическое прогнозирование рассматривается как система научных исследований количественного и качественного характера, которые направлены на выявление тенденций развития экономических отношений и поиск оптимальных решений для достижения поставленных целей.

Экономический прогноз как итог экономического прогнозирования представляет собой научно обоснованное суждение о состояниях экономического объекта в будущем. Такое суждение носит вероятностный характер, однако обладает определенной степенью достоверности.

Экономический прогноз позволяет решить ряд задач, которые лежат в основе планирования деятельности любой фирмы.

Такими задачами могут быть:

- оценка состояния и осуществление поиска возможных вариантов управленческих решений;
- определение очертания области и возможности для изменения будущих событий;
- выявление проблем, слабо выраженных в настоящем, но возможных, в будущем;
- осуществление поиска вариантов активного воздействия на объективные факторы будущего;
- моделирование вариантов событий с учетом главных факторов.

Для отдельной фирмы, как правило, речь может идти о трех следующих видах прогноза.

1. Прогноз в соответствии с проблемно-целевым критерием, т.е. поисковый и нормативный прогнозы, которые дают ответ на вопрос: "С какой целью разрабатывается прогноз?".

Поисковый прогноз (трендовый) помогает определить возможные состояния явления или процесса в будущем. Примером такого прогнозирования может служить прогноз продажи товаров предприятий на ближайшие год - два. Основным методом такого прогнозирования является *экстраполяция* - распространение выводов, полученных из наблюдений над одной частью явления, на другую его часть.

Нормативный прогноз (программный, целевой) выполняется для определения путей и сроков достижения поставленных целей. Например, составляется прогноз динамики численности персонала предприятия при имеющихся условиях экономического развития. Основным методом прогнозирования в этом случае является *интерполяция* — нахождение по ряду данных значений функции ее промежуточных значений.

2. Прогнозы, выделяемые по *критерию природы объекта:* социальные, ресурсные, научно-технические, общественные и личные.

3. Прогнозы, выделяемые по *критерию времени:* оперативные - на период до одного месяца, краткосрочные - от двух месяцев до года, среднесрочные - от одного года до пяти лет, долгосрочные - от пяти до пятнадцати лет и дальнесрочные - свыше пятнадцати лет.

2. Методы прогнозирования экономических параметров

Методы научного экономического прогнозирования включают совокупность принципов, приемов и показателей, используемых для получения прогноза.

Идею теории прогнозирования составляют ее принципы. Такими принципами являются *системность, непрерывность, вариантность, достоверность и эффективность.*

Системность - это множество закономерно связанных друг с другом элементов (явлений, процессов), представляющее собой единое целое.

Непрерывность - это прогнозирование с корректировкой по мере поступления новых данных.

Вариантность предполагает разработку нескольких вариантов прогноза.

Достоверность — это точность и обоснованность прогноза.

Эффективность прогнозирования определяет превышение экономического эффекта от использования прогноза над затратами по разработке.

Приемы прогнозирования представляют собой совокупность математических или логических операций, используемых для получения конкретного прогноза, например, сглаживание или выравнивание динамического ряда, расчет средневзвешенного значения величин.

К наиболее распространенным методам прогнозирования относятся:

- экстраполяция;
- нормативные расчеты (интерполяция);
- экспертные оценки;
- аналогии;
- математическое моделирование.

Экстраполяция по сути является переносом закономерностей и тенденций прошлого на будущее на основе анализа взаимосвязей показателей динамического рядя. Этот метод эффективен для краткосрочных прогнозов.

При *нормативном методе* прогнозирования определяются пути и сроки достижения поставленных целей в развитии явления или процесса. С использованием этого метода можно определить состояние явления или процесса на периоды, предшествующие достижению указанной нормы. Такие расчеты называют *интерполяцией.*

Метод *экспертных оценок* используется в основном при долгосрочном прогнозировании. Прогнозирование осуществляется на основе высказываний высококвалифицированных специалистов - экспертов.

Метод *аналогии* предполагает перенос знаний об одном явлении или процессе на другой. Этот метод в своей основе опирается на абстрактное представление о процессах и явлениях, т. к. сходство между ними редко бывает полным.

Самым сложным методом прогнозирования является *математическое моделирование* — описание явления или процесса с помощью инструментария математики: формул, уравнений, неравенств.

Модель - это явление или процесс, который находится в некотором соответствии с изучаемым явлением или процессом. Модели могут быть материальными и идеальными.

В экономике рассматриваются *идеальные модели,* в частности - математические модели. С помощью таких моделей можно определить:

- зависимости между различными экономическими параметрами;
- ограничения, которые накладываются на экономические параметры;
- критерии, позволяющие оптимизировать процесс.

3. Прогнозирование исходных данных по заданному значению результата

Демонстрационный пример

Довольно часто в экономике необходимо решить задачу как бы "с конца": по заданному значению результата рассчитать значение исходных данных. Например, известен размер вклада, который будет помещен на определенный срок под заданный процент. Требуется определить коэффициент наращивания (т.е. значение, показывающее, во сколько раз увеличится вклад за указанный срок) и сумму выплат в конце периода. Такая задача не представляет особенного труда для ее решения:

Коэфф. наращивания = $(1+\text{проц. ставка})^{\text{срок выполнения}}$

сумма выплат = коэфф. наращивания • размер вклада

Задача несколько усложняется, если требуется определить сумму вклада, исходя из заданной суммы выплат. Решить такую задачу можно лишь опытным путем, применив серию повторений, переборов - *итераций* - различных сочетаний результата и исходной величины до тех пор, пока не будет подобран необходимый вариант. При решении подобных задач различной степени сложности количество итераций может достигать нескольких тысяч.

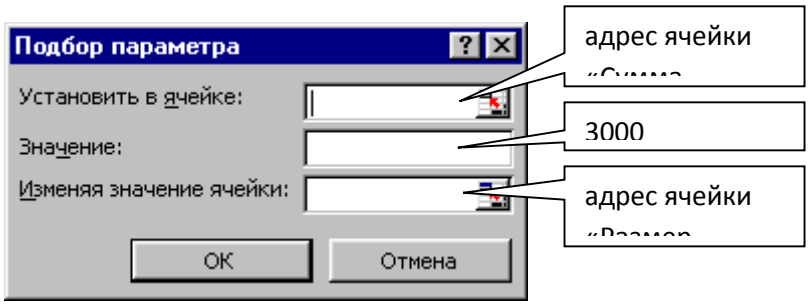
Процессор электронных таблиц MS Excel позволяет достаточно быстро и точно решить подобную задачу. С этой целью используется функция *подбора параметра.*

Для решения такой задачи необходимо иметь таблицу, в которой все расчеты выполнены со ссылками - по формулам.

Сумма выплат по банковским вкладам

Размер вклада (руб.)	Срок вклада (лет)	Процентная ставка (%)	Коэффициент наращивания	Сумма выплат (руб.)
2000	3	1,5	1,05	2091,36

Далее необходимо выполнить команду ***Сервис - Подбор параметра.*** При этом на экране монитора появится диалог, представленный на рисунке.



Диалог функции ***Сервис - Подбор параметра***

К примеру, необходимо подобрать сумму вклада, чтобы через три года сумма выплат составила 3000 рублей. Для этого в поле ***Установить в ячейке*** нужно ввести адрес ячейки со значением суммы выплат. В поле ***Значение*** ввести значение желаемой суммы выплат, в нашем случае это 3000 рублей. В поле ***Изменяя ячейку*** ввести адрес ячейки со значением размера вклада. В результате выполнения функции ***Подбор параметра*** результат будет выведен в виде следующей таблицы.

Сумма выплат по банковским вкладам

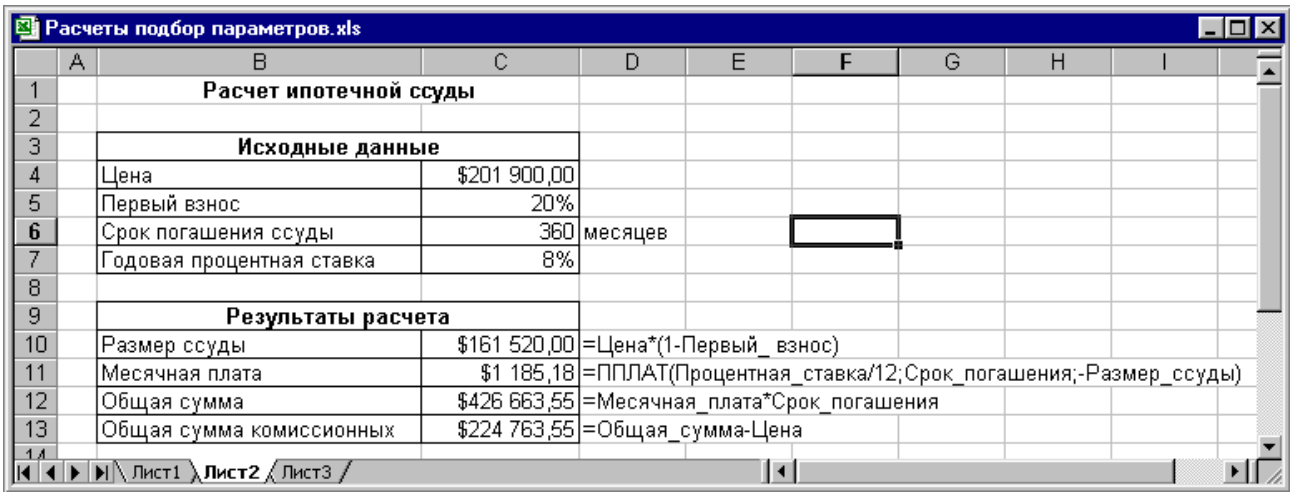
Размер вклада (руб.)	Срок вклада (лет)	Процентная ставка (%)	Коэффициент наращивания	Сумма выплат (руб.)
2868,951	3	1,5	1,05	3000

Таким образом, для того, чтобы по истечении трех лет сумма выплат с заданной процентной ставкой составила 3000 рублей, размер вклада должен составлять 2868,95 рублей.

Задание для самостоятельного выполнения

Задание 1. Расчет и подбор показателей ипотечной ссуды

1. Введите электронную таблицу, которая обрабатывает информацию об ипотечной ссуде. Формулы для расчета приведены в столбце D.



Расчеты подбор параметров.xls

	A	B	C	D	E	F	G	H	I
1		Расчет ипотечной ссуды							
2									
3		Исходные данные							
4		Цена	$201 900,00						
5		Первый взнос	20%						
6		Срок погашения ссуды	360	месяцев					
7		Годовая процентная ставка	8%						
8									
9		Результаты расчета							
10		Размер ссуды	$161 520,00	=Цена*(1-Первый_ взнос)					
11		Месячная плата	$1 185,18	=ПЛАТ(Процентная_ставка/12;Срок_погашения;-Размер_ссуды)					
12		Общая сумма	$426 663,55	=Месячная_плата*Срок_погашения					
13		Общая сумма комиссионных	$224 763,55	=Общая_сумма-Цена					

Лист1 Лист2 Лист3

2. С помощью данной таблицы ответьте на следующие вопросы :

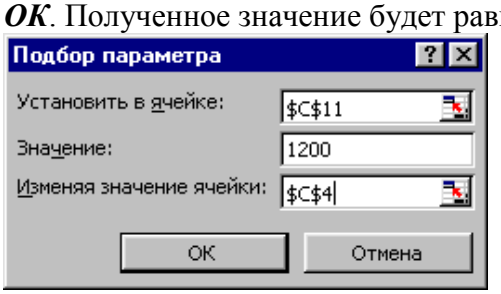
- Каков будет размер ссуды, если я смогу договориться о более низкой цене на имущество ($180 500)?
- Какова будет общая сумма, если я смогу получить 40-летнюю ссуду?
- Какова будет сумма комиссионных, если процентная ставка снизится до 7,5% ?

Каждый новый расчет производите с исходной таблицей. Ответы оформите под таблицей.

Представьте, что вы хотите купить себе новый дом. Вам известно, что в месяц вы в состоянии погашать не более $1 200 взятой ссуды. Вы также знаете, что кредитор даст вам ссуду под фиксированный процент (скажем, 8,25%), рассчитывая на то, что вы погасите за определенное время 80% ссуды (т.е. первоначальный взнос составляет 20%).Вопрос состоит в следующем: «Какова максимальная стоимость покупки, которую вы себе сможете позволить?» Другими словами, какое значение должно быть в ячейке С4, чтобы результат в ячейке С11 равнялся $1 200? Один способ решения – изменять значение в ячейке С4 до тех пор, пока значение в ячейке С11 не станет равным 1200. Более эффективный способ – позволить Excel самому найти ответ.

3. Скопируйте предыдущую таблицу на следующий лист и измените исходные данные (процентную ставку) по заданию.

4. Чтобы ответить на вопрос задачи, выберите команду ***Сервис-Подбор параметра***. Заполните появившееся диалоговое окно. Значения можно набирать вручную или щелкать мышью по нужным ячейкам. Чтобы начать процесс подбора параметра, щелкните на кнопке ***ОК***. Полученное значение будет равно $199 663.



Диалоговое окно «***Подбор параметра»***

Задание 2. Графический подбор параметра

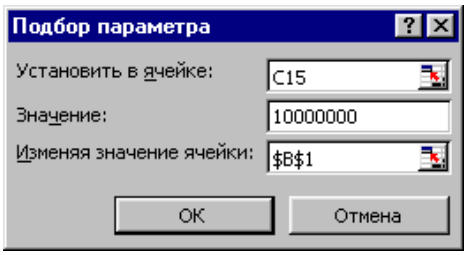
1. На рисунке показан рабочий лист, отображающий предполагаемый объем продаж развивающейся компании. Предположим, что из опыта известно, что рост объема продаж компаний, работающих в этой отрасли, может увеличиваться по закону: =y*(b^x), где y – константа, равная росту продаж за первый год; b – коэффициент роста; x – переменная, выражающая время.

Введите исходные данные, рассчитайте продажи по приведенной формуле (в формуле продаж для разных строк должны сохраняться две константы: y и b, поэтому необходимо применить абсолютные ссылки), постройте гистограмму.

	A	B	C
1	Коэффициент	1,4	
2	Первый год	$250 000	
3			
4		Год	Продажи
5	0	2000	$250 000
6	1	2001	$350 000
7	2	2002	$490 000
8	3	2003	$686 000
9	4	2004	$960 400
10	5	2005	$1 344 560
11	6	2006	$1 882 384
12	7	2007	$2 635 338
13	8	2008	$3 689 473
14	9	2009	$5 165 262
15	10	2010	$7 231 366
16			
17			
18			

2. Менеджеры компании знают, что объем продаж за первый год будет составлять $250 000 и хотят довести его к 2010 г. до $10 000 000. Таким образом, для построения финансовой модели нужно знать точный коэффициент роста, ведущий к заданному объему продаж. Предположительный коэффициент не дал нужного результата.

Щелкните по столбикам диаграммы, а затем щелкните по последнему столбцу. Поместите указатель курсора на вершину столбика и перетащите ее вверх. Когда значение станет равным 10 000 000, опустите кнопку мыши.



3. В появившемся окне введите значение, изменяющейся ячейки и нажмите ОК.

Цель работы: научиться строить прогноз на основе динамических рядов, используя линию тренда.

Задание. Прочитайте теоретические сведения, выполните демонстрационные примеры, выполните задания для самостоятельной работы.

Методические указания:
Прогнозирование на основе анализа временных рядов

Экстраполяция в той или иной форме довольно широко используется управляющими фирм, экономистами, исследователями рынка и всеми, кто занимается прогнозированием. Как метод прогнозирования экстраполяция может включать различные процедуры, в том числе и проектирование трендов. Типичным для экстраполяционных методов является то,

что они тесно не связаны с экономической теорией, однако в разумных пределах удовлетворяют требованиям внешне - экономической деятельности при прогнозировании.

Простейшие модели экстраполяции - это неразрывные модели, т.е. в них все будущие значения изучаемой переменной каким-либо образом являются функцией ее настоящего или недавнего прошлого состояния. Такими простейшими моделями могут быть:

- *не изменяющиеся* модели; в этом случае прогнозируемое значение переменной y^p для следующего периода равняется ее значению в настоящий период y;

- *пропорционально – изменяющиеся модели;* в этом случае прогнозируемое значение переменной y^p будет пропорционально изменению значения переменной y в прошлом.

Такие модели применяются в основном для построения краткосрочных прогнозов, являются наиболее легко осуществимыми, так как они просты и не требуют большого количества информации для расчетов.

Временные ряды состоят из значений, соответствующих определенным точкам или периодам. Тенденция изменения или долговременное увеличение или уменьшение ряда представляет собой *тренд.*

Как метод прогнозирования проектирование тренда чаще всего предполагает, что тенденция изменения переменной продолжится в будущем. На этом основаны принципы прогнозирования тренда с использованием регрессионного анализа. Наиболее распространенным методом выявления тренда является метод наименьших квадратов: подбор линии регрессии таким образом, чтобы сумма квадратов их отклонений от линии регрессии была минимальной:

Процессор электронных таблиц Excel позволяет подобрать одно из пяти видов уравнений парной регрессии: линейная; полиномиальная; логарифмическая ; экспоненциальная; степенная.

Оценить достоверность тренда позволяет *коэффициент детерминации* - R^2. Если $R^2 \geq 0,5$, то можно считать, что выбранное уравнение парной регрессии достаточно точно аппроксимирует исследуемый временной ряд.

Демонстрационный пример

Например, на основании данных импорта товаров за предыдущие пять лет необходимо дать прогноз на следующие два года.

Объемы импорта товаров по периодам

	1	2	3	4	5
Объемы импорта обуви (млн. руб.)	947,5	1012,5	1145,8	1139,7	1156,4

Предполагая, что тенденция изменения импорта товара продолжится в будущем, можно использовать проектирование тренда с помощью моделей парной регрессии, предлагаемых в MS Excel. Для этого необходимо после формирования таблицы исходных данных построить диаграмму типа «График».

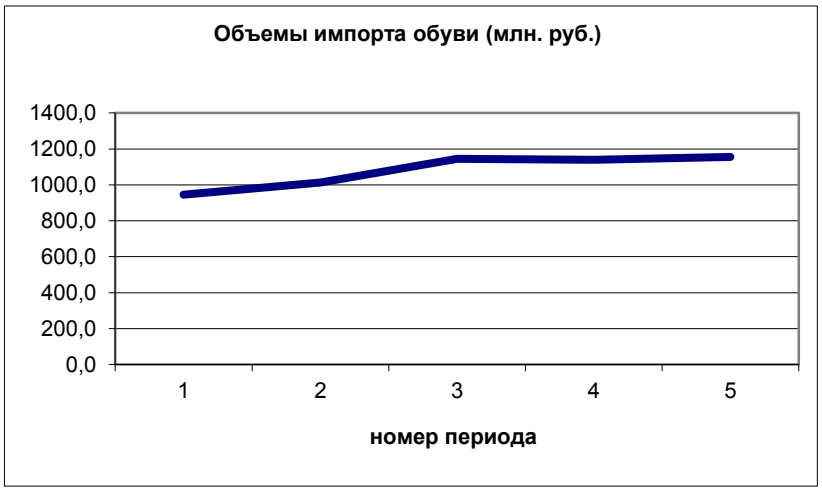
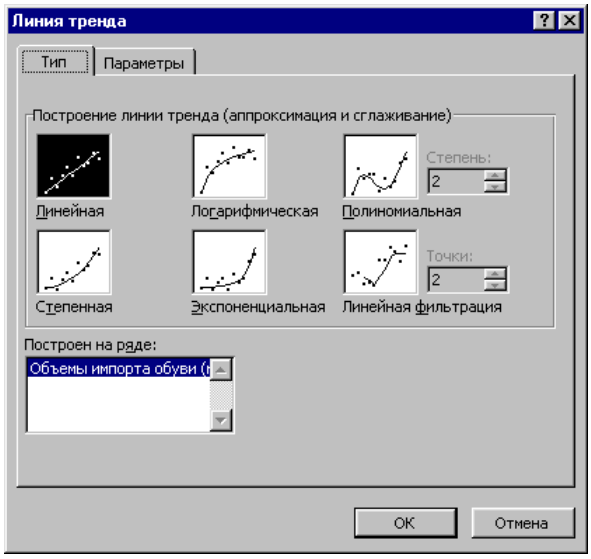


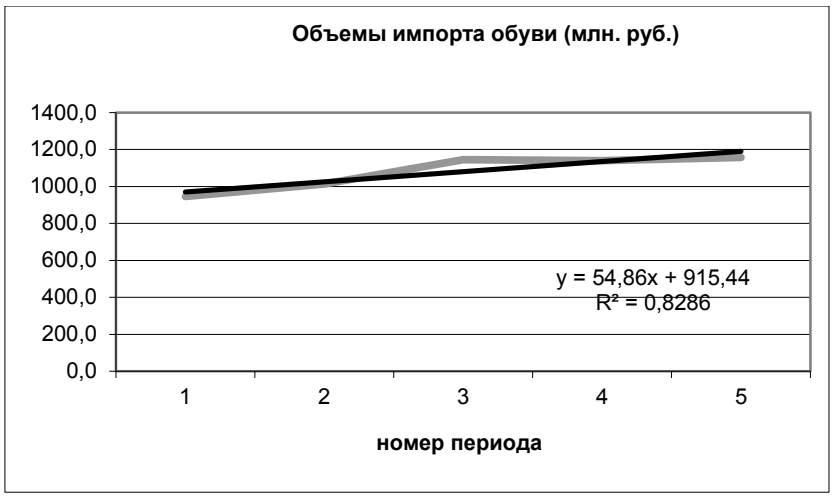
Диаграмма импорта товаров

Далее щелкнув мышью на линии графика (выделив ее), необходимо выполнить команду ***Диаграмма – Добавить линию тренда***. При этом на экране монитора появиться диалог, представленный на рисунке.



Диалог функции ***«Добавить линию тренда»***

Для подбора уравнения парной регрессии щелкнуть мышью на одной из предложенных моделей, перейти на вкладку ***Параметры*** и установить в развернувшемся окне «флажки» для вывода на экран монитора уравнения регрессии и значения коэффициента детерминации.



Линия тренда на диаграмме

Задание для самостоятельного выполнения

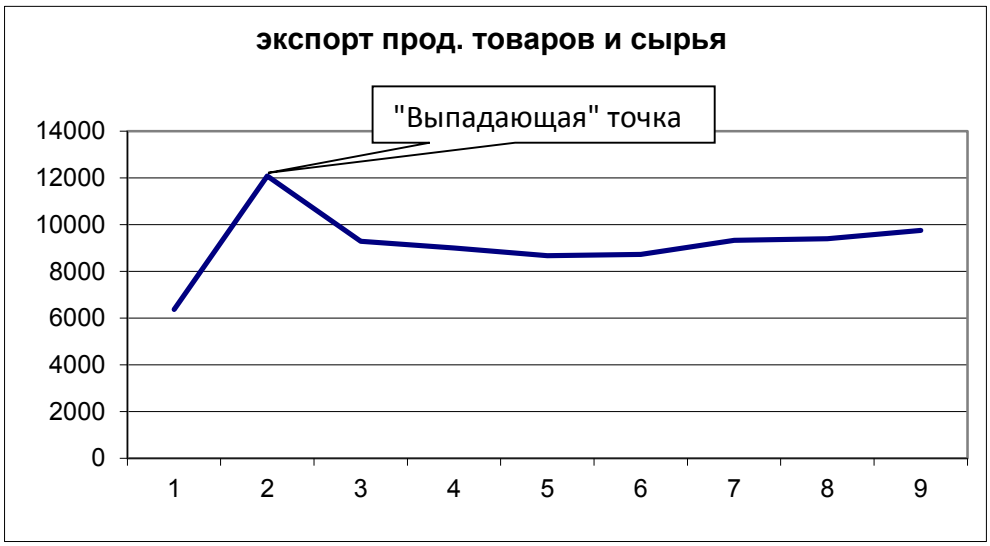
Задание 1. Прогнозирование на основе динамических рядов

1. На рабочем листе текущей рабочей книги сформировать таблицу с исходными данными: объемами экспорта/импорта товаров с 1994 по 2002 годы в страны СНГ. Форма таблицы и необходимые исходные данные представлены в таблице ниже.

Товарная структура экспорта/импорта в страны СНГ
1999 - 2008 годы (тыс. дол. США)

№ п/п	Наименование товара	1999	2000	2001	2002	2003	2004	2005	2006
		1	2	3	4	5	6	7	8
1	Продовольственные товары и сырье для их производства								
	экспорт	6367,03	12083,3	9284,99	8997,14	8664,25	8720,93	9324,09	9395,21
	импорт	17564,21	15269,05	19875,45	19658,3	17458,4	17015,5	16985,5	16125,32

2. По данным экспорта продовольственных товаров и сырья для их производства (диапазон С6:К6) построить диаграмму типа ***«График».*** Результат такого построения приведен на рисунке.



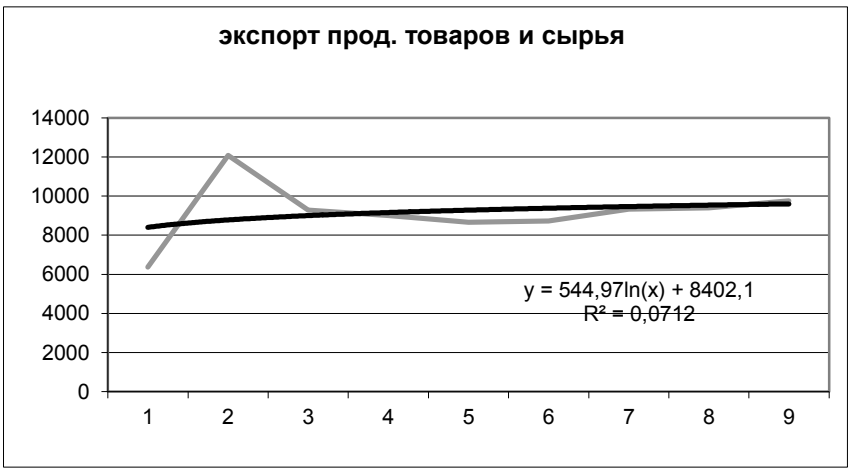
Динамика экспорта продовольственных товаров и сырья

3. Используя функцию MS EXCEL ***«Линия тренда на диаграмме»,*** подобрать уравнение парной регрессии, наиболее точно аппроксимирующее динамический ряд данных по экспорту/импорту товаров. Для этого выполнить следующие действия:

3.1. Выделить линию графика на диаграмме щелчком левой кнопки мыши.

3.2. Выполнить команду меню ***Диаграмма — Добавить линию тренда.***

3.3. В развернувшемся диалоге выбрать одну из моделей парной регрессии; для вывода на плоскость диаграммы уравнения парной регрессии и оценки точности аппроксимации (коэффициента детерминации) активизировать вкладку ***Параметры*** и выставить метки в соответствующих полях.



Линия тренда на диаграмме экспорта продовольственных товаров и сырья

На рисунке приведен пример результата выбора логарифмической парной регрессии, из которого видно, что значение коэффициента детерминации (R^2) равно 0,0712. Это говорит о том, что выбранное уравнение логарифмической парной регрессии не может быть использовано для расчета прогноза на последующие годы.

3.4. Провести анализ причин полученного результата; в нашем примере причиной такого низкого значения достоверности аппроксимации $R^2 = 0{,}0712$ является наличие точки резко «выпадающей» из общей тенденции — значение объема экспорта товара во втором

периоде - 1995 год; это можно объяснить воздействием каких-либо случайных факторов, не оказывающих существенного влияния на динамику ряда в дальнейшем. В таких случаях используют прием «усреднения», т.е. вместо такого «выпадающего» значения берется среднее арифметическое значение между значением переменной в предыдущем и последующем периодах.

3.5. Ввести в ячейку *D6* формулу для расчета среднего значения: *=(C6+E6)/2*.

3.8 Исходя из значения коэффициента достоверности аппроксимации $R^2 = 0,8043$, можно сделать заключение о возможности применения уравнения парной логарифмической регрессии для расчета прогноза экспорта товаров на последующие периоды.

4. Рассчитать значение прогноза экспорта продовольственных товаров и сырья для их производства на последующие три периода (добавьте эти периоды в таблицу).

4.1. В ячейку *L6* ввести уравнение парной логарифмической регрессии, полученное при вставке линии тренда; в качестве значения аргумента *X* используется значение ячейки *L4* - номер периода; в нашем случае введенная формула будет выглядеть следующим образом: *=1295,1 *Ln(L4)+6862*

4.2. Скопировать введенную в ячейку *L6* формулу на весь требуемый диапазон для расчета прогноза (в нашем случае это ячейки *M6 и N6)*.

4.3. Выполнить пункты 2 - 4.2 данного задания для импорта товаров.

Скопировать все полученные ранее результаты (таблицы, графики, ответы на вопросы) в текстовый редактор Microsoft Word и оформить отчет о работе, указав свои Ф.И.О. и группу. Для каждого практического задания ввести заголовок. Вывести на печать полученный результат.

Практическая работа №7
Проектирование баз знаний с использованием средств табличного процессора

Цель работы: сформировать навыки проектирования баз знаний с использование табличного процессора, сформировать умения по созданию базы знаний в табличном процессоре.

Оборудование, технические и программные средства: персональный компьютер, табличный процессор **MS Excel**.

Теоретическая часть

База данных (date base) – это совокупность хранимых в памяти компьютера данных, которые отображают состояние некоторой предметной области. Данные взаимосвязаны и специальным образом организованы.

При таком информационном отображении предметных сред упор делается не на сами объекты и их свойства, а на отношения между ними, что соответствует так называемой реляционной точке зрения на базы данных.

Excel умеет складывать, вычитать, умножать, делить и выполнять множество других операций. Excel дает возможность предварительно проанализировать последствия принятия тех или иных решений при конкретных обстоятельствах. Excel позволяет автоматизировать не только расчеты как таковые, но позволяет создавать и работать с разнообразными картотеками, системами учета, базами данных и т.п.

Вопросы сбора данных, их хранения, учета и обработки можно решить, имея систему управления списками. Термин **список** используется в Excel для обозначения **базы данных.**

База данных – это особый тип рабочей таблицы, в которой не столько вычисляются

новые значения, сколько размещаются большие объемы информации в связанном виде.

Например, можно создать базу данных с фамилиями, именами, адресами и номерами телефонов ваших знакомых или список группы со всей информацией об итогах сессии и о размере соответствующей стипендии или ее отсутствии.

База данных представляет собой последовательность **записей**, содержащую однозначно определенную по категориям и последовательности информацию. Под каждую категорию данных в записи отводится отдельное **поле**, которому присваивается имя и отводится столбец.

Задание 1. Создать базу данных «Студенты».

Методические указания по выполнению задания:

1. Запустите табличный процессор **MS Excel**.
2. Создайте в папке своей группы новую папку и назовите её **Колледж**. Сохраните базу данных «Студенты», выполнив команду **Файл – Сохранить как…**
3. Заполните базу данных и оформите как на рис.

	A	B	C	D	E	F	G	H
1	Фамилия	Имя	Отчество	Высшая математика	Информатика	Физика	Ср. бал	Стипендия
2	Иванов	Иван	Иванович	4	5	3	4	12

Примечание. В ячейки D2:F2 таблицы вводятся оценки, а в ячейку G2 – средний балл. Стипендия вычисляется по формуле: если Ср.балл меньше 4, стипендия не начисляется (=0); если Ср.балл больше 4, но меньше 4.5, начисляется стипендия 12 $., а если больше 4.5, то – 15$.

В столбцы «Ср.балл» и «Стипендия» введены формулы:

Ср.балл=СРЗНАЧ(E2:H2);

Стипендия=ЕСЛИ(G2<4;0;ЕСЛИ(G2<4,5;12;15))

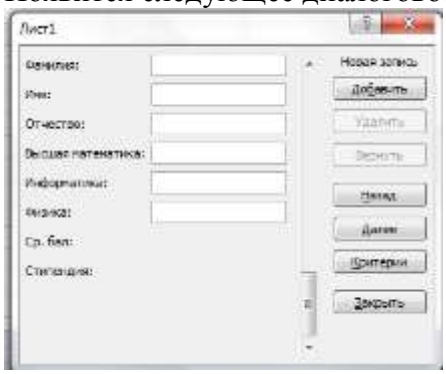
4. Добавьте данные с помощью Формы для этого необходимо выбрать команду **Данные – Форма.**

Если кнопки **Форма** нет, то можно добавить на панель быстрого доступа.

Добавление кнопки "Форма" на панель быстрого доступа

1. Щелкните стрелку, расположенную рядом с **панелью быстрого доступа**, и выберите пункт **Другие команды.**
2. В поле **Выбрать команды из** выберите пункт **Все команды.**
3. В списке выберите кнопку **Форма** и нажмите кнопку **Добавить.**

Появится следующее диалоговое окно



Чтобы ввести новую запись, нужно щелкнуть на кнопке **Добавить** и ввести данные в пустые поля. Для перехода к очередному полю следует нажимать клавишу <**Tab**> .

Добавьте еще 10 строк в базу данных при помощи Формы

5. Найдите Иванова, воспользовавшись Формой – Критерии (Введя необходимый параметр). Для этого необходимо:

- Щелкнуть на первой записи .
- Выбрать **Данные – Форма.**
- Щелкнуть на кнопке **Критерии** в появившейся форме, после чего очистятся все поля формы.
- Ввести требуемые критерии поиска в соответствующие поля формы.
- Нажать кнопку **Далее** или клавишу **Enter** , чтобы начать поиск.
- Excel отобразит форму данных с первой найденной записью, а чтобы просмотреть все следующие , следует нажимать кнопку **Далее**; при необходимости пройтись по найденным записям в обратном порядке следует щелкать по кнопке **Назад.**

6. Отредактируйте пятого студента поменяв Фамилию на Сергеев(а).

Для **редактирования** значения поля в текущей записи необходимо перейти в него, нажимая клавиши <**Tab**> или <**Shift+Tab**> (или с помощью мыши) , и ввести новое значение. Для очистки поля целиком выделить его и нажать клавишу <**Del**>.

7. Удалите восьмую запись.

Для **удаления** записи из БД щелкнуть на кнопке **Удалить** в окне формы данных. При этом, однако, следует помнить, что невозможно восстановить удаленную таким образом запись с помощью команды **Отменить**. Поэтому Excel выдаст окно предупреждения с таким сообщением:« Запись, выведенная на экран, будет удалена ». Можно подтвердить свое решение об удалении записи, щелкнув на кнопке **ОК** , или отменить, щелкнув на кнопке **Отмена .**

8. Упорядочите данные воспользовавшись сортировкой по следующим полям:

- Средний балл – по возрастанию;
- Стипендия – по убыванию.

9. Упорядочите данные воспользовавшись Фильтром. Для активизации нужного фильтра следует выполнить такие действия:

- Щелкнуть где-либо в БД.
- Выбрать **Данные – Фильтр – Автофильтр** , в результате чего Excel добавит раскрывающийся список к каждой ячейке с именем поля в строке заголовков.
- Щелкнуть на кнопке списка поля, по значению которого нужно выполнить отбор записей , и выбрать один из предложенных вариантов фильтра, щелкнув по нему.

После этого Excel покажет только записи , содержащие в этом поле выбранное значение (все остальные будут временно скрыты). Отфильтрованные записи можно скопировать на другой лист рабочей книги или выдать на печать. Для этого нужно просто:

- отметить клетки;
- выбрать команду **Копировать** из меню **Правка** (или нажать <**Ctrl+C**>);
- переместить табличный курсор в первую ячейку таблицы на новом листе, щелкнув по ярлычку «Лист2» в нижней части экрана;
- нажать **Enter**.

Для вывода на печать отфильтрованных записей достаточно после их выделения щелкнуть на кнопке **Печать** на инструментальной панели или выбрать команду **Печать** в меню **Файл** .

10. Отфильтруйте БД по полю стипендия и значению 15$.

11. Дополнительно к фильтрации БД по записям, содержащим определенное значение в поле, можно создавать собственные автофильтры, позволяющие фильтровать БД по записям с более общим критерием, таким как, например, фамилии, начинающиеся с буквы «А», или значения средних баллов в пределах от 4 до 5. Для создания собственного

фильтра нужно:

- щелкнуть на кнопке раскрывающегося списка в названии поля ;
- выбрать опцию **Условие**;
- в появившемся диалоговом окне **Пользовательский автофильтр** выбрать необходимый оператор сравнения в первой строке или в обеих строках, если условие составное, т.е. представляет собой результат логических операций типа «и» и «или»; в текстовые окна справа ввести значения (текст или число), относительно которых должно проводиться сравнение значений в записях БД.

Так для выбора списка студентов, фамилии которых начинаются с буквы «А» необходимо в первой строке диалогового окна **Пользовательский автофильтр** щелкнуть «равно» и ввести в текстовое окно «А*» (без кавычек).

Для отбора только студентов, имеющих средний балл в пределах от 4 до 5 , нужно задать условие : «больше или равно» 4 «и» «меньше или равно» 5 (в кавычках – операторы, которые следует выбрать, а 4 и 5 нужно набрать в текстовых полях) .

12. Создайте сводную таблицу воспользовавшись **Мастером сводных таблиц**.

- Выполнить команды **Данные —Сводная таблица** .
- В первом диалоговом окне выбрать диапазон данных, где расположена таблица.
- Нажать кнопку Далее, в результате чего появится второе диалоговое окно, в котором необходимо выбрать данные, которые необходимо вывести. Вывести Фамилию, Имя, Ср.балл, Стипендию

13. Excel предоставляет возможность подвести промежуточные и окончательные **итоги** по полям БД, вставив необходимые строки для подсчитанных сумм . Кроме того, Excel может отображать БД в режиме структуры, что позволяет выводить на экран только промежуточные итоги.

Посчитайте итоги по стипендии и Ср.баллу, воспользовавшись **Данные – Промежуточный итог.**

Задание 2. Проектирование базы данных «Бригада»

На этапе проектирования базы данных задаем структуру базы, определяем количество, наименование и типы полей базы, определяем для вычисляемых данных формулы, по которым они вычисляются.

1. Структура базы данных

В соответствии с заданием база данных должна содержать следующие поля:

№ поля	Имя поля	Тип поля	Тип данных	Длина поля
1	ФИО	Символьный	Исходные	20 символов
2	Бригада	Символьный	Исходные	12 символов
3	Специальность	Символьный	Исходные	15 символов
4	Оклад	Числовой	Исходные	4 символа
5	Премия	Числовой	Вычисляемые	4 символа
6	Начислено	Числовой	Вычисляемые	4 символа
7	Налог	Числовой	Вычисляемые	8 символов
8	К выплате	Числовой	Вычисляемые	8 символов

Рис. Структура проектируемой базы данных.

2. Определение формул для вычисляемой части базы данных

В создаваемой базе несколько вычисляемых полей Определим зависимости, по которым вычисляются значения в этих полях. Введем условные обозначения, которые будем использовать при составлении формул:

Премия – П;

Оклад – О;

Стаж- С;

Начисленная сумма - НС;

Подоходный налог - ПН;

1.Премия.

В соответствии с условиями премия начисляется сотрудникам, проработавшим определенное время на фирме. Для сотрудников со стажем от 2-х до 5-ти лет премия составит 15% оклада, со стажем более 5 лет 25% оклада.

$$\prod = \begin{cases} 0 & если\ C \le 2 \\ 0.15 * O & если\ 2 < C \le 5 \\ 0.25 * O & если\ C > 5 \end{cases}$$

При использовании Мастера функции логическое выражение для вычисления премии приобретет вид:

П= Если (С <= 2; 0; если(С>5;0.25*О;0.15*О))

2.Начисленная сумма. Значение начисленной суммы определяется как результат сложения значения оклада и премии.

НС=П+О.

3.Подоходный налог.

Подоходный налог определяется в зависимости от величины начисленной суммы: не облагаются налогом суммы до 70 грв. включительно; при начисленной сумме более 250 грв. подоходный налог составляет 20% от суммы; в остальных случаях подоходный налог равен 10% от суммы.

$$ПН = \begin{cases} 0 & если\ \ HC \le 70 \\ 0{,}1 * HC & если\ \ 70 < HC \le 250 \\ 0{,}2 * HC & если\ \ HC > 250 \end{cases}$$

При использовании Мастера функций логическое выражение примет вид:

ПН=Если(НС<=70;0;если(НС>250;0.2*НС;0,1*НС))

1. К выплате. Значение определяется как разность Начисленной суммы и Подоходного налога.

К_вып=НС-ПН

В результате проектирования базы данных получен эскиз таблицы следующего вида.

	A	B	C	D	E	F	G	H	I
1	ФИО	Бригада	Специальность	Оклад	Стаж	Премия	Начисленная сумма	Подоходный налог	К выплате
2									

Рис. 2.2. Эскиз таблицы для заполнения базы данных.

В таблице исходные данные отмечены синим цветом, а вычисляемые значения – красным.

3. Создание базы данных

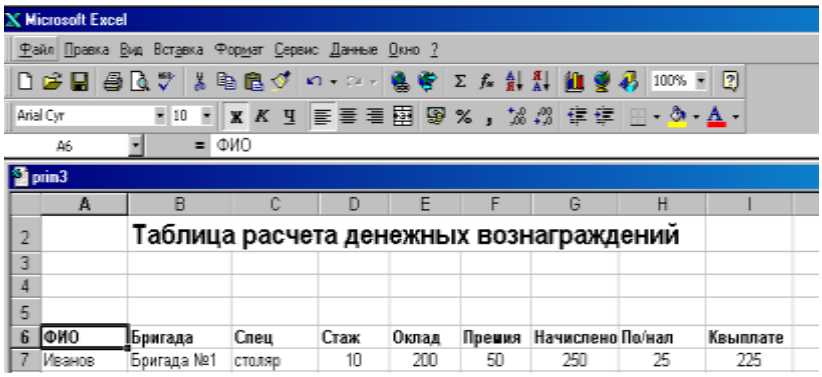


Рис. 2.3. Заголовок таблицы.

4. Заполнение таблицы с помощью Мастера форм

Дальнейшее заполнение данных выполняем с помощью пунктов меню **Данные Форма** (в этот момент курсор должен быть установлен на ячейке в области создаваемой таблицы). При этом открывается окно диалога Мастера форм с заполненной первой

записью исходной базы данных.

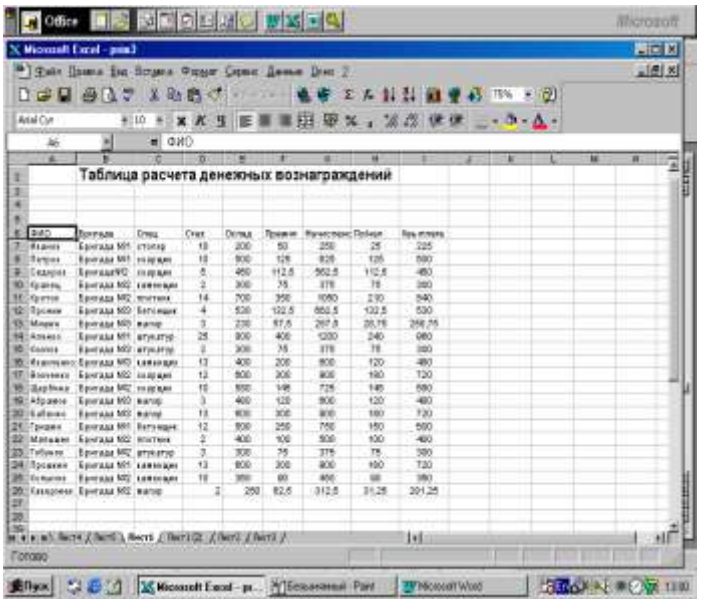
Рис.2.4 Окно диалога Мастера форм.

Щелкая на клавише «Добавить» в окне диалога и последовательно заполняя пустые поля исходными данными, создаем исходную базу данных. Окончательный вид базы данных приведен на Рис.2.5.

Рис.2.5. Исходная база данных.

5. Ведение базы данных

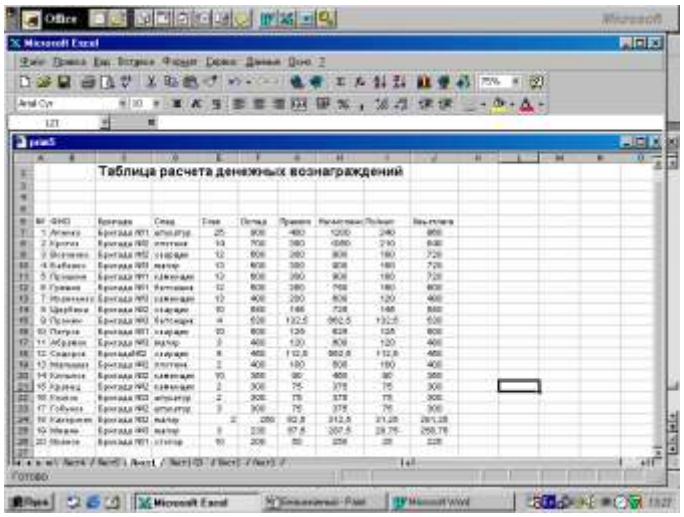
Ведение базы данных заключается в корректировке существующих данных, добавлении новых, удалении полей, содержащих ненужную или ошибочную информацию.



В соответствии с заданием выполним операции по редактированию, добавлению, удалению информации. При работе в среде электронных таблиц для этого могут быть использованы различные средства.

6. Редактирование полей

Добавим к существующей базе поле, которое отражает порядковый номер записей в базе. Для этого установим курсор в ячейку первого столбца и активизируем пункт меню **Вставка Столбцы**. В появившемся пустом столбце запишем название поля и заполним его.



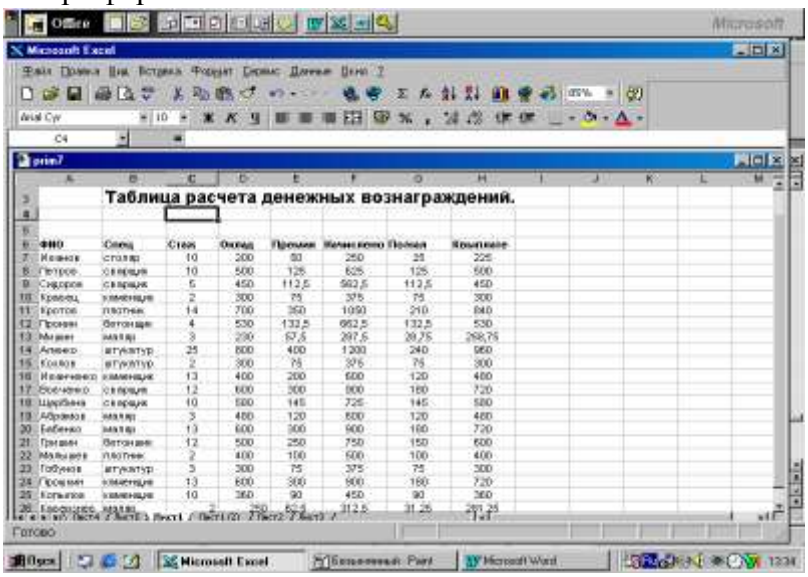
Результат поместим в новый файл. Удаление полей осуществим с помощью команд меню **Правка Удалить Удалить столбец**. При этом курсор необходимо установить в поле , которое собираемся удалить. На Рис. 2.6 показан вид базы данных с добавленным полем №, а на Рис.2.7 показана база данных, из которой удалили поля № и «Бригада».

7. Редактирование записей

Для удаления записей из базы данных необходимо эти записи выделить и активизировать команды меню **Правка Удалить Строку .** В результате выполнения этих действий строка, в которой был установлен курсор будет удалена. Для добавления строк в

базу данных необходимо выполнить следующие действия: активизировать команды меню **Вставка Строки.** В результате в базу данных будет добавлена пустая строка над строкой, в которой находился курсор. Далее заносим нужные сведения в добавленную строку. Рис. База данных с удаленными полями № и “Бригада”

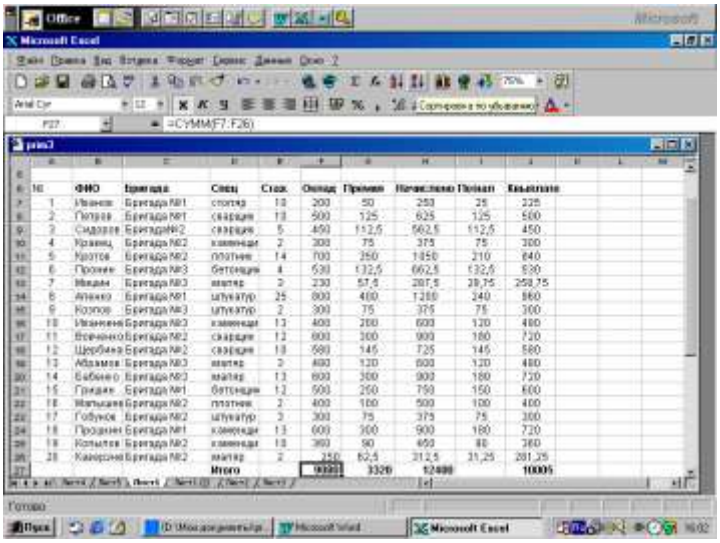
В ячейке сохранится откорректированная информация. Добавим к исходной базе строку, содержащую информацию о рабочем Васечкине и исправим фамилию Иванов на Иванченко в 1-й записи. При редактировании можно также пользоваться окном диалога Мастера форм.



8. Добавление суммы по столбцам

Добавим в числовых полях суммирование по столбцам.

Рис. Добавление суммирования по столбцам.



Практическая работа №8

Сортировка базы данных

Цель работы: сформировать навыки проектирования баз знаний с использование табличного процессора, сформировать умения по созданию базы знаний в табличном процессоре.

9. Сортировка базы данных

Для обычной сортировки базы данных по одному полю (по возрастанию или по

убыванию) необходимо воспользоваться пунктом меню **Данные Сортировка** или пиктограммами на графическом меню.

Сортировка по возрастанию Сортировка по убыванию

При этом курсор должен быть установлен в поле, которое будем сортировать. Для сортировки по возрастанию по полю «Стаж» установим курсор на ячейку в этом поле и выберем направление сортировки «по возрастанию». Результат сортировки представим на Рис.

Таблица расчета денежных вознаграждений

№	ФИО	Бригада	Спец	Стаж	Оклад	Премия	Начислен	По/нал	Квыплате
1	Малышев	Бригада № 2	плотник	2	400	100	500	100	400
2	Кравец	Бригада № 2	каменщик	2	300	75	375	75	300
3	Козлов	Бригада № 3	штукатур	2	300	75	375	75	300
4	Каверзнев	Бригада № 2	маляр	2	250	62,5	312,5	31,25	281,25
5	Абрамов	Бригада № 3	маляр	3	480	120	600	120	480
6	Гобунов	Бригада № 2	штукатур	3	300	75	375	75	300
7	Мишин	Бригада № 3	маляр	3	230	57,5	287,5	28,75	258,75
8	Пронин	Бригада № 3	бетонщик	4	530	132,5	662,5	132,5	530
9	Сидоров	Бригада№ 2	сварщик	5	450	112,5	562,5	112,5	450
10	Щербина	Бригада № 2	сварщик	10	580	145	725	145	580
11	Петров	Бригада № 1	сварщик	10	500	125	625	125	500
12	Копылов	Бригада № 2	каменщик	10	360	90	450	90	360
13	Иванов	Бригада № 1	столяр	10	200	50	250	25	225
14	Вовченко	Бригада № 2	сварщик	12	600	300	900	180	720
15	Гришин	Бригада № 1	бетоншик	12	500	250	750	150	600
16	Бабенко	Бригада № 3	маляр	13	600	300	900	180	720
17	Прошкин	Бригада № 1	каменщик	13	600	300	900	180	720
18	Иванченко	Бригада № 3	каменщик	13	400	200	600	120	480
19	Кротов	Бригада № 2	плотник	14	700	350	1050	210	840
20	Аленко	Бригада № 1	штукатур	25	800	400	1200	240	960

Рис. Сортировка по полю «Стаж» по возрастанию.

10. Сортировка по нескольким полям

Для проведения более сложной сортировки (по нескольким полям) откроем окно диалога «Сортировка диапазона» (см. Рис.2.13). Для этого выполним команды меню **Данные Сортировка.** Чтобы сортировка выполнялась по двум или трем полям в окне диалога для каждого диапазона задаем направление сортировки. Наименование диапазонов выбираем в окне диалога , раскрывая список наименований (щелкаем последовательно по областям окна диалога «Сортировать по», «Затем по», «В последнюю очередь по»), и указываем направление сортировки по каждому полю (убывание/возрастание). В нашем примере выполним сортировку по трем критериям: по полю «Стаж», затем по полю «Специальность» и в последнюю очередь по полю «Оклад».Для всех трех критериев задаем направление сортировки «По возрастанию».

Таблица расчета денежных вознаграждений

№	ФИО	Бригада	Спец	Стаж	Оклад	Премия	Начислено	По/нал
15	Гришин	Бригада № 1	бетоншик	12	500	250	750	150
18	Прошкин	Бригада № 1	каменщик	13	600	300	900	180
2	Петров	Бригада № 1	сварщик	10	500	125	625	125
1	Иванов	Бригада № 1	столяр	10	200	50	250	25
8	Апенко	Бригада № 1	штукатур	25	800	400	1200	240
4	Кравец	Бригада № 2	каменщик	2	300	75	375	75
19	Копылов	Бригада № 2	каменщик	10	360	90	450	90
20	Каверзнев	Бригада № 2	маляр	2	250	62,5	312,5	31,25
16	Малышев	Бригада № 2	плотник	2	400	100	500	100
5	Кротов	Бригада № 2	плотник	14	700	350	1050	210
12	Щербина	Бригада № 2	сварщик	10	580	145	725	145
11	Вовченко	Бригада № 2	сварщик	12	600	300	900	180
17	Гобунов	Бригада № 2	штукатур	3	300	75	375	75
6	Пронин	Бригада № 3	бетонщик	4	530	132,5	662,5	132,5
10	Иванченко	Бригада № 3	каменщик	13	400	200	600	120
7	Мишин	Бригада № 3	маляр	3	230	57,5	287,5	28,75
13	Абрамов	Бригада № 3	маляр	3	480	120	600	120
14	Бабенко	Бригада № 3	маляр	13	600	300	900	180
9	Козлов	Бригада № 3	штукатур	2	300	75	375	75
3	Сидоров		сварщик	5	450	112,5	562,5	112,5

Рис.2.14. Результат сортировки по трем полям.

11. Добавление промежуточных итогов.

Добавление промежуточных и окончательных итогов выполняется после сортировки исходной базы по выбранному полю. Выполним сортировку исходной базы по полю «Бригада» и добавим промежуточные и общий итоги. Для добавления итогов откроем окно диалога «Промежуточные итоги». Для этого выполняем команды меню **Данные Итоги.** В окне диалога зададим поле, в котором будет отслеживаться изменение значений (например поле «Бригада»). Затем укажем в строке «Операция» тот вид операции, который нужен для выполнения задания(Сумма, Среднее и т.д.). В списке окна диалога «Добавить итоги по» укажем, каких еще полях базы данных необходимо выполнить аналогичные действия.

12. Построение диаграммы, отражающей премию сотрудников

Практическая работа №9
Цифровая обработка изображений

Выполнив данную практическую работу, вы:

- Узнаете как обработать изображение для публикации в Интернет или отправке по электронной почте.
- Научитесь создавать ScreenServer (заставку рабочего стола).
- Научитесь переименовывать файл или группу файлов, конвертировать файл или группу файлов из одного формата в другой.

Обработка фотографий с помощью каталогизатора изображений ACDSee Systems является отличной альтернативой для тех, кто не может подружиться с Фотошопом. На начальном этапе для подготовки изображений к публикации в Интернет или отправке по почте достаточно графического редактора ACDSee 5.0. Тем более, что в 5-ой версии появилось много инструментов из Фотошопа.

В расширенный пакет ACDSee Systems входят:

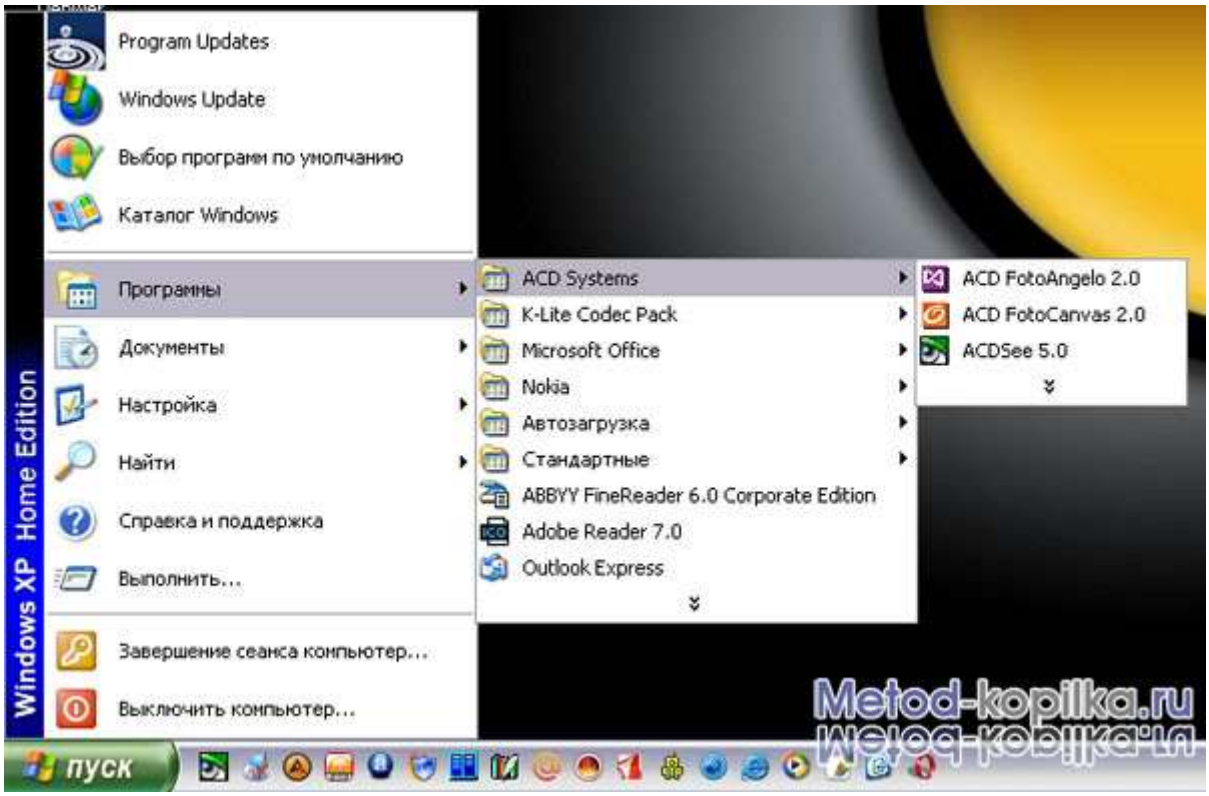
1. ACD FotoAngelo 2.0

2. ACD FotoCanvas 2.0

3. ACDSee 5.0

Технология выполнения работы:

1. Пуск - Программы - ACDSee Systems

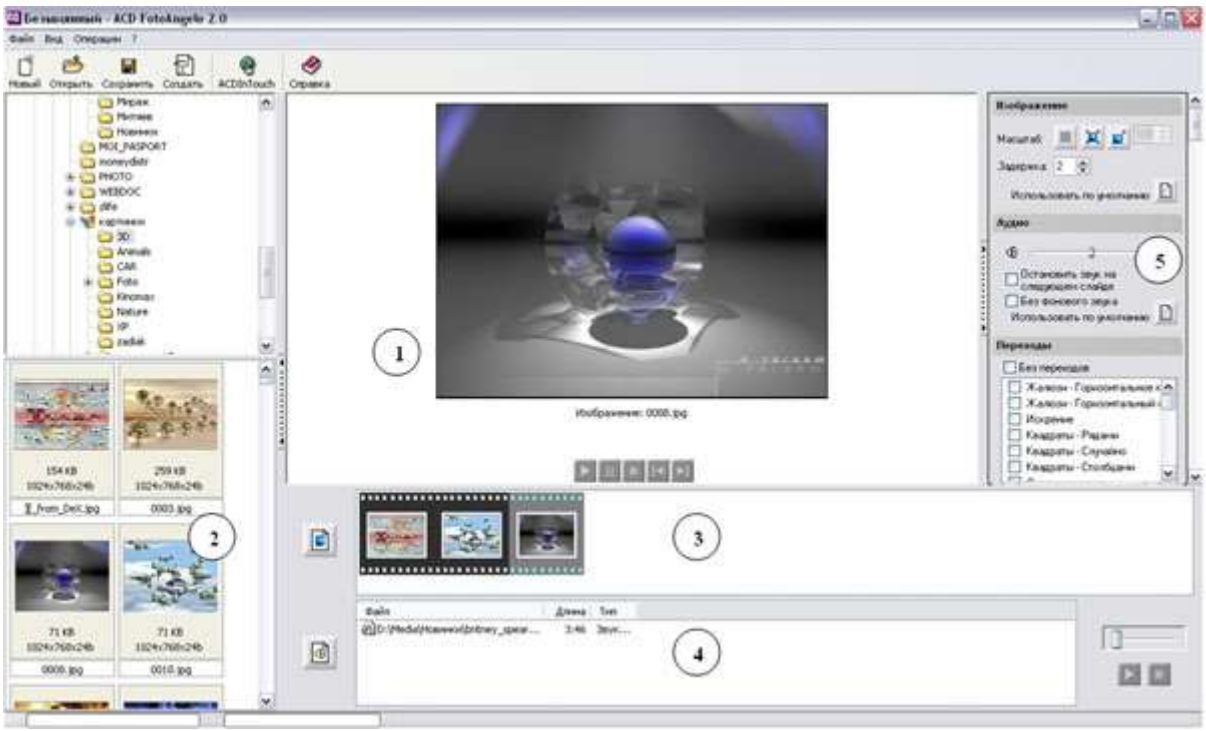


2. Запустим **ACD FotoAngelo 2.0**. В окне "Быстрое начало работы" предложены 3 режима работы: Создать новый проект, Открыть имеющийся проект или открыть один из предыдущих проектов.

ACD FotoAngelo 2.0
Простая и удобная программа для создания слайд-шоу (и скринсейверов - слайд-шоу). Можно встраивать звук, текст, выбирать эффекты замены изображений. Сама по себе поддерживает 13 графических форматов, а если у вас уже установлена ACDSee, то более 40.

Задача: Создать ScreenServer (заставку рабочего стола). Мы создадим **Новый проект**.

1. В левой колонке - каталог папок и файлов найдем папку, содержащую необходимые изображения **(2).**



2. Перетащим изображения из каталога **(2)** - в окно слайдов **(3)**, аналогично выбираем звуковой файл **(4).**

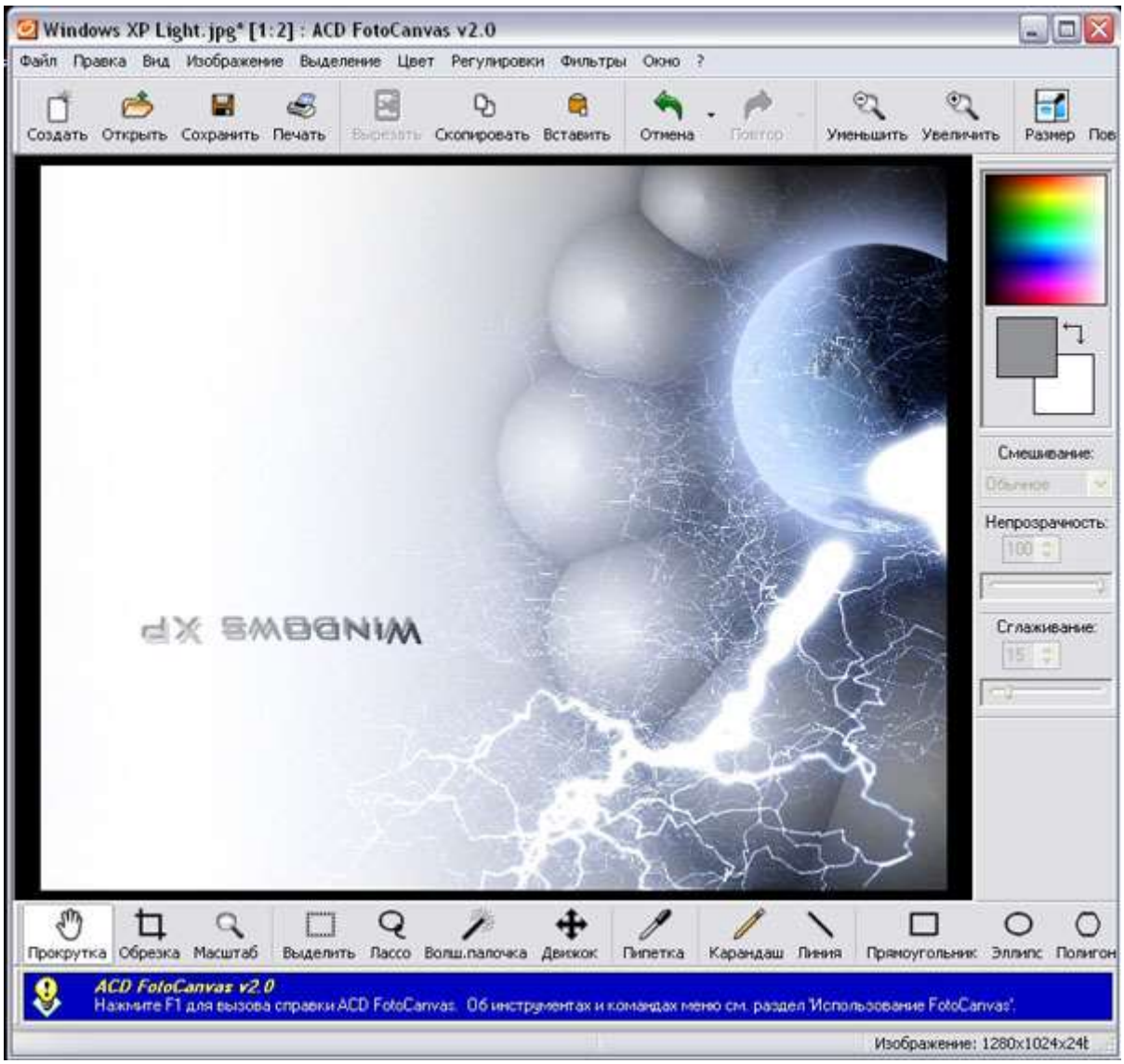
3. На панели настройки **(5)** настроим параметры изображения, аудифайлов, выберем вид перехода от слайда к слайду.

4. Сохраним созданый проект в виде заставки, в главном меню выберем**Операции - Создать** - экранная заставка (.scr) - выберите разрешение заставки под размеры вашего экрана, задайте имя для вашего файла.

ACD FotoCanvas 2.0

Простая и удобная программа для редактирования изображений.

1. Отредактируем изображение средствами **ACD FotoCanvas 2.0** Для этого можно открыть изображение, выбрав в контектном меню **Правка** или в главном меню **Действия - Правка - Редактор.**

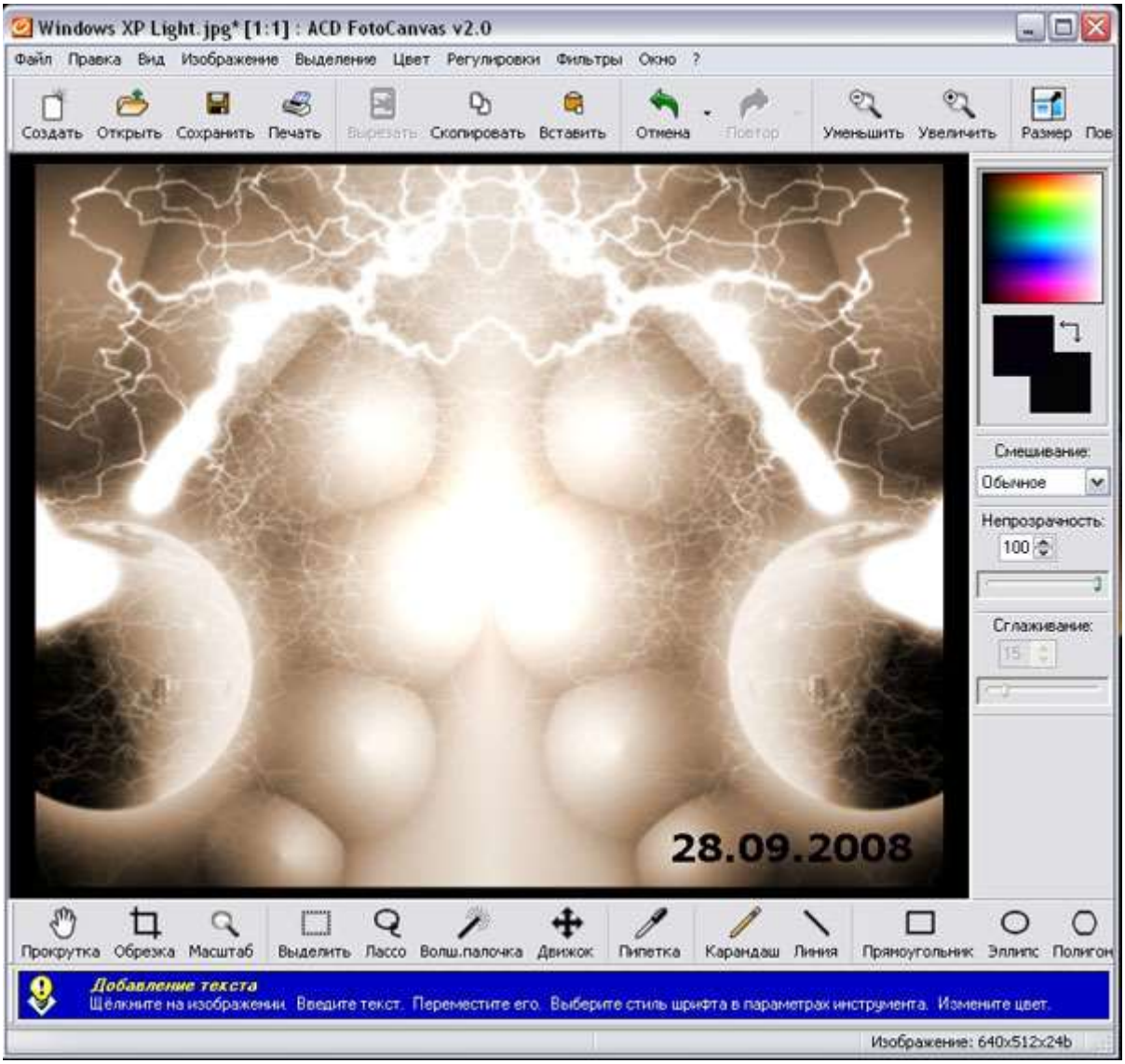


2. Изображение перевернуто, развернем его. Сделать это можно с помощью кнопок **Повернуть** или **Перевернуть**. Поэксперементируйте.

3. **Фильтры.** Встроенные инструменты фильтры (аналог Photoshop).

- Состарим картинку. Фильтр **Цвет-Сепия.** Поиграйте с настройками.
- Отобразим изображение в зеркально. **Фильтр-Деформация-Зеркало.**Поиграйте с настройками.
- Разместим световое пятно - солнце в центре картинки. **Фильтр-Освещение-Солнце**. Установите солнце в необходимом месте.
- Фильтр-Художественные-Перекрестная штриховка.
- 4. Откорректируем цвет картинки - Автоуровни. Поиграйте с настройками.
- 5. Изменим размер картинки. **Размер** - Процент от исходного - 50%. Кроме этого в **ACD FotoCanvas 2.0** есть возможность задать размер по пикселям, сохраняя или нет пропорции.
- 6. Добавим на нашу картинку текстовый блок с датой редактирования рисунка. Щелкаем по кнопке **Текст** и выбрав тип шрифта, размер, начертание и выравнивание на сой вкус, впишите дату. Чтобы переместить дату по картинке, необходимо

подвести указатель мыши к текстовому блоку, курсор превратится в четырехнаправленную стрелку, удерживая левую кнопку перемещаем блок.



- **Задача.** Подготовим фото к публикации на сайте.
- ***Технология выполнения работы:***
- 1.Полученный файл "Портрет.tiff" открываем в программе просмотра (ACDSee), и сразу переходим в "Редактор" (Foto Canvas).

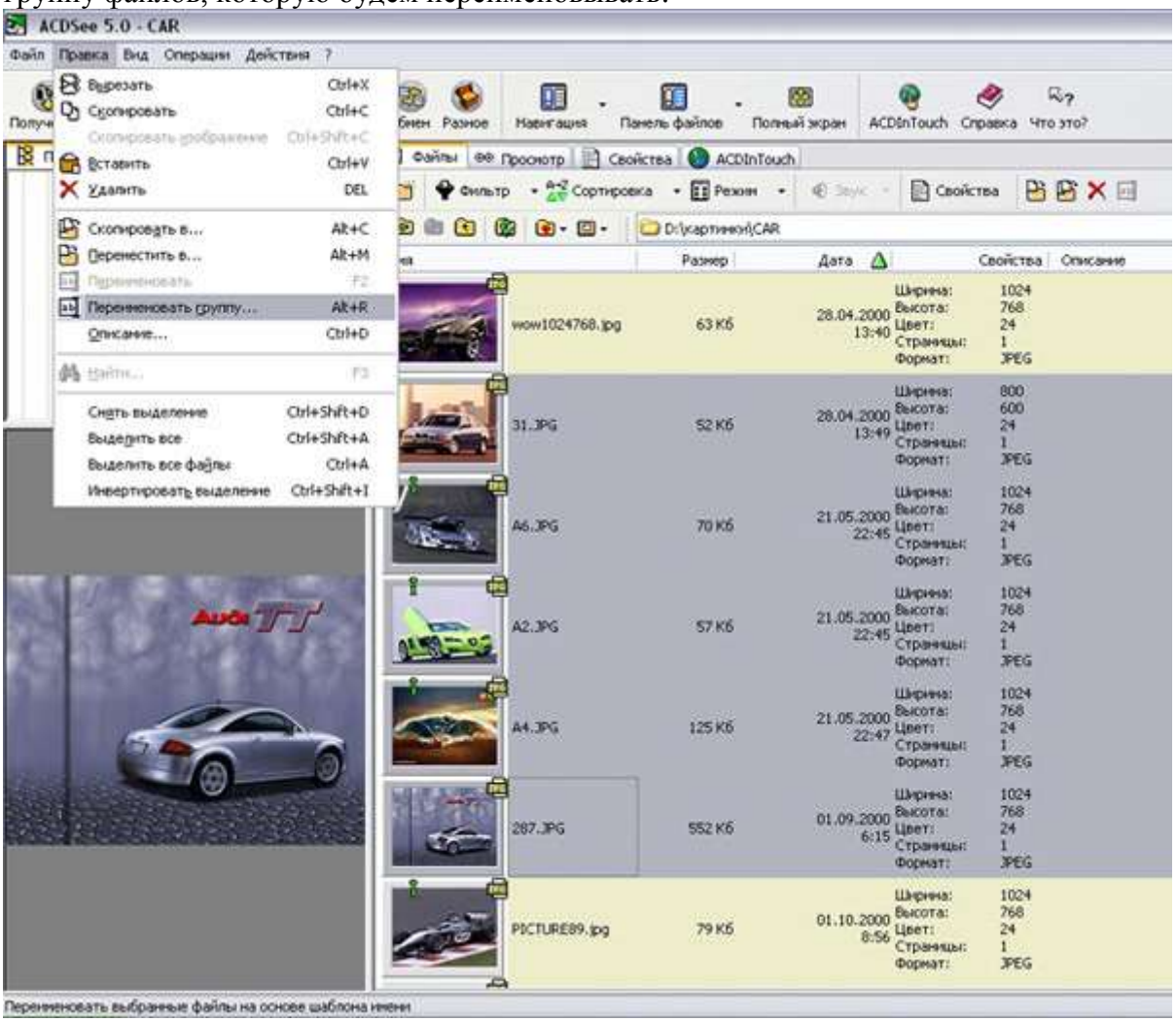
- 2.Сначала кадрируем (обрезаем) изображение чтобы выделить сюжетно важную часть снимка. Кадрировать важно и нужно.
- Резать нужно без жалости, убирая бесполезные и мешающие детали.**Инструмент - Вид - Инструменты рисования - Обрезка**. Обрезаем до подходящего размера растягивая и сужая границы инструмента, и подтверждаем двумя подряд нажатиями левой кнопки мыши внутри обрезаемой части изображения. Не нравится - **Отмена.** Повторяем эти операции до приемлемой кадрировки.
- 3.Пробуем применить **Автоуровни**, иногда это даёт неплохой результат или помогает понять, что нужно делать дальше. Если не устраивает результат жмём **Отменить.**
- 4.Заходим на **Регулировки - Изменить**. Играем с "Чёрным" и "Белым".
- 5.Потом **Регулировки- Тон/Нас/Светлота**,корректируем, если требуется. Иногда полезно чуть-чуть убрать **Насыщенность.**
- 6.Следом **Регулировки- Красный/Зел/Синий** - корректируем, иногда довольно долго, чтобы добиться хорошего результата.
- 7.Если требуется подавляем шум через **Шумоподавление.** Но этот инструмент не настраиваемый, а работает уж слишком эффективно, в результате получаются "кукольные" лица. Поэтому я чаще пользуюсь**Фильтры - Шум - Удаление медианного шума**. Так как при уменьшении размера шум уходит сам собой, почти всегда, оставляю как есть.
- 8.Теперь уменьшаем размер. Внимание. Приведённые ниже размеры относятся к необрезанному файлу. Будем считать, что пункт 2(кадрировка) мы не выполняли. Правило 50% процентного пошагового уменьшения действительно в любом случае.
- Всегда следует уменьшать размер не более чем в 50% от исходного.
- Итак. Размер - 50%
- 9.Резкость, естественно, упала. Повышаем её **Фильтры- Резкость**. Здесь надо осторожно выбрать от 50 до 90 процентов, потому что ближе к 90 % появляется, так

называемая, "зашарпленность", изображение становится плоским и появляются некрасивые контурные границы белого цвета.

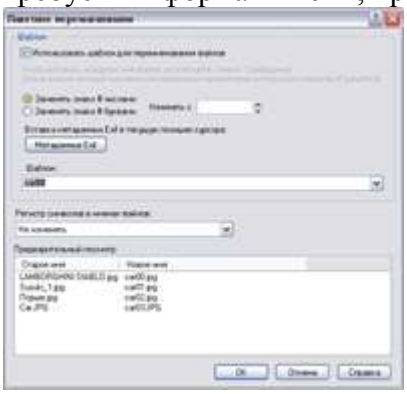
- 10.Слегка упала яркость изображения. Можно добавить **Регулировки-Тон/Нас/Светлота**, пару делений в **Светлота**. Чаще полезнее добавить**Регулировки- Экспозиция -Белый.**
- Теперь главное - как всю эту красоту сохранить с минимальными потерями в *jpg - формате.
- 12.Файл "Портрет.tiff" - сохраняем через **Файл- Сохранить как...** и в названии ставим дополнительный значок - "Портрет1.tiff", чтобы не перезаписать исходный файл. Дело в том, что сразу в *jpg переводить не следует - почему-то сильно падает качество, да и неудобно.
- 13.Поэтому переходим обратно в программу просмотра и находим отредактированный файл "Портрет1.tiff" (Горизонтальный - около 850 кб, вертикальный - около 430 кб.) **Правой кнопкой - контекстное меню - Преобразовать**, выбираем JPG-JPEG.
- Регулируем "Параметры формата" экспериментально. Моя задача, как правило, не прывысить 100 кб при хорошем качестве.
- В редких случаях я допускаю превышение размера до 140 кб для сохранения максимального качества. Вертикальный кадр позволяет сохранять до 100% качества при размере до 100 кб. Горизонтальный - от 60 до 90 %.
- Можно сравнить с исходным файлом "Портрет.tiff" весом в 26 мегабайт.

- **ACDSee 5.0**
  Простая и удобная программа для просмотра, пермещения, копирования, переименования изображений ограниченного количества форматов.
- В **Проводнике** открываем необходимый файл или папку. В меню **Правка**размещены следущие очень удобные функции **Скопировать в..., Переместить в...**

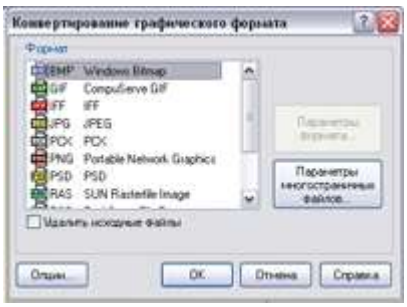
- Рассмотрим полезную команду **Переименовать группу**, предварительно выделяем группу файлов, которую будем переименовывать.



- Открывается окно **Пакетное переименование**. В окне **Шаблон** можно ввести требуемый формат имени, причем, выбрать возможность замены знака #.



- Довольно часто нам приходится изображения конвертировать в тот или иной необходимый формат. Сделать это можно выбрав в главном меню пункт **Операции - Конвертирование формата.**
- В опциях конвертирования формата изображения можно измененные изображения разместить в исходную папку или в другую выбранную папку, также можно удалить исходные файлы.



Практическая работа №10
Цифровая обработка видео
Цель: освоить навыки работы при создании и обработке видео файлов.
Задание: изучить технологию создания и обработки видео файлов в Movie Maker.
Форма контроля: проверка электронных файлов.
Создание клипов
Для упрощения работы над проектом можно создать из одного видеоклипа несколько легко обрабатываемых клипов меньшего размера. Windows MovieMaker создает клипы различными способами в зависимости от их источника. Если источник клипа – цифровая камера, Windows MovieMaker создает клипы на основе меток времени, добавленных цифровой видеокамерой при записи, а также на основе существенных изменений видеокадра.
1. На панели содержимого выберите видеоклип, для которого необходимо создать клипы.
2. Щелкните Сервис, а затем щелкните Создать клипы (рис. 1).

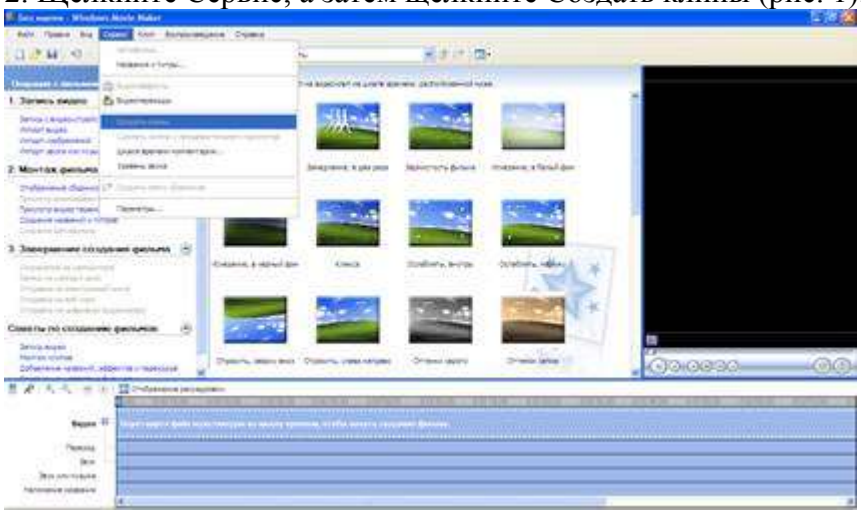


Рис. 1. Создание клипа
Примечание
Для видеофайлов формата WMV (Windows MediaVideo) и AVI (Audio-VideoInterleaved), использующих цифровой видеокодек, клипы могут создаваться автоматически. Для других форматов видеофайлов клипы не всегда могут создаваться автоматически, поэтому в программе Windows MovieMaker видеофайл будет представлен как один большой видеоклип. Эти большие видеоклипы придется разделить на клипы меньшего размера вручную.
Можно улучшить процесс создания фильма с помощью добавления собственных оригинальных штрихов, которые придадут фильму особый профессиональный вид. Благодаря переходам и эффектам ваш фильм будет плавно перетекать от одной сцены к другой, и выглядеть именно так, как вам требуется.
Переходы
Переход управляет сменой одного видеоклипа или изображения другим (рис. 2).

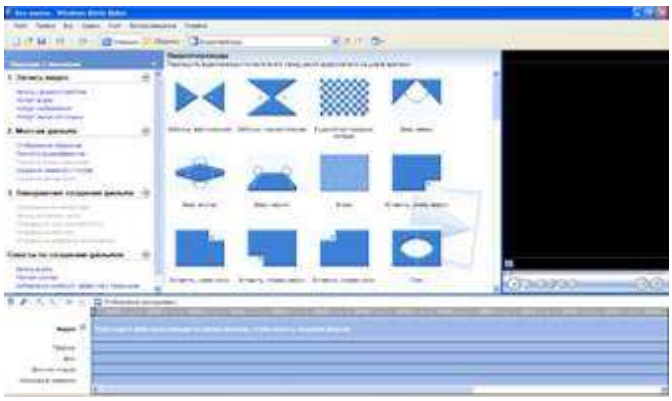


Рис. 2. Переход

Сам процесс перехода называют «монтажом» или «склейкой». Так, в одно целое связываются порой беспорядочные вещи, дающие новую ритмическую последовательность.

С помощью программы Movie Maker можно добавить переход между двумя изображениями, видеоклипами или заголовками в любой комбинации в раскадровке или на шкале времени. Можно выбрать такой популярный и привлекательный переход, как Угасание. Также можно использовать более яркие переходы, такие как Решетка, На части или Зигзаг (и многие другие).

Все добавленные переходы появляются на дорожке перехода шкалы времени. Чтобы увидеть эту дорожку, необходимо растянуть видеодорожку.

Добавление перехода

1. В раскадровке или на шкале времени выберите второй из двух видеоклипов, заголовков или изображений, между которыми необходимо добавить переход.
2. Щелкните Сервис, затем щелкните Переходы.
3. В области содержимого выберите переход, который следует добавить. Для предварительного просмотра перехода можно нажать Воспроизвести под окном просмотра.
4. Щелкните Клип, затем щелкните Добавить на шкалу времени или Добавить на раскадровку.

Примечание

Также можно добавить переход, перетаскивая его на шкалу времени между двумя видеоклипами на видеодорожке. Либо в режиме раскадровки можно перетащить переход в ячейку перехода между двумя видеоклипами или изображениями.

Изменение продолжительности перехода

Продолжительность перехода определяется временем перекрытия между двумя клипами. Иногда необходимо сделать переход меньше или больше.

1. Чтобы просмотреть дорожку перехода шкалы времени, растяните видеодорожку.
2. На дорожке перехода шкалы времени выполните одно из следующих действий.

Для уменьшения продолжительности перехода перетащите начало перехода по направлению к концу шкалы времени.

Для увеличения продолжительности перехода перетащите начало перехода по направлению к началу шкалы времени.

Удаление перехода

1. Выполните одно из следующих действий.

Щелкните в раскадровке ячейку перехода, содержащую переход, который необходимо удалить.

Щелкните на шкале времени переход на дорожке перехода, который необходимо удалить.

Щелкните Правка, затем щелкните Удалить.

Эффекты

Эффекты позволяют добавлять к фильму спецэффекты. Например, можно придать импортированному видео вид классического, старого фильма. Для этого можно добавить к видеоклипу, изображению или заголовку один из эффектов фильма под старину, чтобы клип выглядел как старый фильм.

Добавление эффекта

Рис. 3. Добавление эффекта

1. На раскадровке или шкале времени выберите видеоклип, изображение или заголовок, к которому необходимо добавить эффект (рис. 3).
2. Щелкните Сервис, затем щелкните Эффекты.
3. В области содержимого выберите эффект, который следует добавить. Можно нажать Воспроизвести под монитором, чтобы осуществить предварительный просмотр эффекта.
4. Щелкните Клип, затем щелкните Добавить на шкалу времени или Добавить на раскадровку.

Примечания

Также можно добавить эффект, перетаскивая его из области содержимого на изображение или видеоклип на видеодорожке шкалы времени, в ячейку эффекта видеоклипа или на изображение на раскадровке.

Режим эскизов в области содержимого отображает примеры различных эффектов.

Изменение эффекта

1. На видеодорожке шкалы времени или на раскадровке выберите видеоклип, изображение или заголовок, к которому применен эффект, который требуется изменить.
2. Щелкните Клип, выберите Видео, затем щелкните Эффекты.
3. Выполните одно из следующих действий.

Чтобы удалить эффект, выберите его в области Отображаемые эффекты, затем щелкните Удалить. При необходимости повторите.

Чтобы добавить эффект, выберите его в области Имеющиеся эффекты и щелкните Добавить. При необходимости повторите.

С помощью Windows MovieMaker можно добавлять в фильм название, имя создателя, дату, титры и другой текст. Например, можно добавить титры, чтобы представить человека или сцену в фильме.

На этом уроке мы на практике рассмотрели работу с программой WindowsMovieMaker.

На следующем уроке мы на практике рассмотрим создание презентаций в программе MicrosoftPowerPoint.

Изменение продолжительности перехода по умолчанию

1. Щелкните Сервис, выберите Параметры, затем щелкните вкладку Дополнительно.
2. Введите время (в секундах), в течение которого переходы должны воспроизводиться по умолчанию после их добавления в раскадровку или на шкалу времени.

Советы

Если добавлено несколько эффектов, можно изменить порядок их отображения с помощью кнопок Вверх и Вниз.

Чтобы быстро добавить эффект, можно перетащить его на видеоклип, изображение или заголовок на раскадровке или шкале времени.

Если добавить к клипу один и тот же эффект более одного раза, он будет применен соответствующее количество раз. Например, если дважды добавить к одному и тому же видеоклипу эффект ускорения в два раза, клип будет воспроизводиться в четыре раза быстрее исходного.

Также можно удалить эффект следующим способом: выбрать на раскадровке ячейку с эффектом, который необходимо удалить, а затем нажать клавишу DELETE.

Добавление названия и титров

Рис. 4. Добавление названия

1 Щелкните на раскадровке или шкале времени там, где должны отображаться название или титры в фильме.

2. В меню Сервис щелкните Названия и титры.

3. Щелкните ссылку на место добавления названия или титров.

4. В поле Введите текст названия введите текст названия или титров.

После ввода текста на экране отображаются тип анимации и формат по умолчанию добавляемых названия или титров.

5. Для изменения типа анимации названия щелкните Изменить анимацию названия и выберите в списке тип анимации.

6. Для изменения шрифта и цвета титров щелкните Изменить шрифт и цвет текста и выберите тип шрифта, его цвет, форматирование, цвет фона, прозрачность, размер шрифта и положение названия.

7. Щелкните Добавить название.

Изменение названия

1. На раскадровке или шкале времени выберите изменяемое название.

2. В меню Правка выберите команду Изменить название.

3. Внесите нужные изменения и щелкните Добавить название.

Изменение времени показа титров

1. Для перехода в режим шкалы времени в меню Вид выберите Шкала времени.

2. Выберите титр, время отображения которого следует изменить.

3. Для увеличения времени воспроизведения титров перетащите конечный маркер монтажа в конец шкалы времени.

Для уменьшения времени воспроизведения титров перетащите начальный маркер монтажа в конец шкалы времени.

Удаление титров

1. На раскадровке или шкале времени щелкните удаляемые из фильма титры.

2. В меню Правка щелкните Удалить.

Список литературы

1. Босова Л.Л. Информатика и ИКТ: Учебник для 6 класса. – М.: БИНОМ. Лаборатория знаний, 2012.
2. Босова Л.Л. Информатика: Рабочая тетрадь для 6 класса. – М.: БИНОМ. Лаборатория знаний, 2010.
3. Босова Л.Л., Босова А.Ю. Уроки информатики в 5-6 классах: Методическое пособие. – М.: БИНОМ. Лаборатория знаний, 2010.

Дополнительные рекомендованные ссылки на ресурсы сети Интернет

1. Интернет портал «School.xvatit.com» (Источник).
2. Интернет портал «Nsportal.ru» (Источник).
3. Интернет портал «Lamptrade.su» (Источник).

Домашнее задание

1. Подготовьте небольшое сообщение о программе Windows Movie Maker;
2. Опишите основные позиции интерфейса программы Windows Movie Maker;
3. Создайте небольшой видеоролик в программе Windows Movie Maker.

Практическая работа №11
Решение прикладных задач с помощью статической экспертной системы

Цель работы: научиться проводить онтологические исследования, составлять словарь терминов и список взаимосвязей объектов выбранной ПО;
получить практический опыт построения концептуальной модели знаний ПО; научиться формализовать концептуальную модель знаний в виде правил логического вывода (концептуальная модель должна допускать это);
научиться строить машину вывода (решатель) в виде дерева решений и реализовывать машину вывода с помощью таблицы переходов.

1.1. Краткие теоретические сведения

Система, которую мы намерены построить, относится к классу идентификационных (или диагностических) систем, которые решают задачу идентификации (определения) объекта по его признакам. Такие системы составляют значительную часть существующих экспертных систем, и без их рассмотрения не обходится ни один учебник по экспертным системам.

Мы будем реализовывать следующий план построения экспертной системы. Сначала построим решатель, затем (на следующих лабораторных занятиях) добавим интерфейс пользователя и блок объяснения, что позволит продемонстрировать работу экспертной системы в полном объёме.

Мы будем строить Экспертную Систему реляционного типа, используя дерево решений (машина вывода), отражающее знания и опыт эксперта в решении задач в данной проблемной области. Для реализации машины вывода и получения искомого решения мы будем применять таблицу переходов *.

(*) *Идея реляционной модели Экспертной Системы была предложена американским ученым Е.Ф. Коддом в начале 70-х. Слово "реляционная" происходит от английского*

relation - отношение, связь. Суть реляционного подхода заключается в том, что информация об объектах представляется в виде отношений, т.е. связанных между собой характеристик изучаемых объектов. В свою очередь, отношение удобно представлять в виде таблицы, в которой каждая строка содержит значения характеристик рассматриваемых объектов.

Следует отметить, что построение Экспертной Системы реляционного типа с использованием дерева решений является наиболее простым, но не единственным способом построения диагностической экспертной системы.

В **Приложении Б** приведен альтернативный способ построения экспертной системы – построение Экспертной Системы продукционного типа, в которой используется система правил продукций, отражающих экспертные знания. Дано описание работы решателя (машины вывода) в такой системе при использовании прямого и обратного вывода.

Машину вывода (Решатель) реляционной ЭС мы изобразим орграфом – ориентированным графом, который состоит из точек, называемых вершинами орграфа, и линий со стрелками, соединяющими эти точки. Каждая такая линия называется дугой орграфа.

Каждая вершина орграфа помечена либо уточняющим вопросом экспертной системы к пользователю, либо ответом ЭС на задачу (решение ЭС).

Для удобства все вершины пронумерованы, начиная с нуля.

Если вершина помечена вопросом экспертной системы, то из нее выходят две дуги (в случае многоальтернативного выбора количество дуг соответствует числу возможных выборов). Каждая дуга соответствует одному из альтернативных ответов пользователя. Вершина, соответствующая ответу ЭС на поставленную задачу (решение ЭС), не имеет выходящих дуг.

1.2. Пример построения Экспертной Системы реляционного типа

Рассмотрим, как спроектировать и построить экспертную систему для идентификации объектов в заданной ПО. В нашем примере экспертная система поможет нам выбрать фотоаппарат в соответствии с нашими требованиями, опытом и финансовыми возможностями. Этот пример позволит понять, как строить идентификационную ЭС в любой другой проблемной области.

Для простоты мы ограничимся случаем, когда пользователь может точно ответить на вопрос о наличии или отсутствии того или иного признака (детерминированная экспертная система).

Начнем с того, что сформулируем **знания** по указанному вопросу.

1.3. Факты, свидетельства, гипотезы (декларативные знания)

1. Если ваш бюджет ограничен, и стоимость фотоаппарата имеет для вас существенное значение, мы предлагаем вам простые фотоаппараты-мыльницы
2. Если ваш бюджет ограничен и для вас важно качество фотографий, то Вам подойдут фотоаппараты мыльницы с отличным качеством снимков, стоимостью выше 7-8 тыс. руб.
3. Если ваш бюджет ограничен и для вас важно наличие видео съемки, вам подойдут такие фотоаппараты, как Nikon Coolpix S3100 Red и проч.

4. Если вы не ограничены в средствах, то вам могут быть доступны профессиональные фотоаппараты.
5. Если вы не ограничены в средствах, и у вас нет опыта работы с профессиональными фотоаппаратами, то предлагаем вам купить полупрофессиональные фотоаппараты.
6. Если вы не ограничены в средствах, и у вас есть опыт работы с профессиональными фотоаппаратами меньше года, то предлагаем вам купить китовые фотоаппараты - это профессиональные фотоаппараты со стандартным объективом.
7. Если вы не ограничены в средствах, и у вас есть опыт работы с профессиональными фотоаппаратами больше года, вы можете приобрести фотоаппараты с объективом, отличающимся от стандартного.
8. Если вы не ограничены в средствах, у вас есть опыт в использовании профессиональных фотоаппаратов больше года, и вы предпочитаете фотографировать внутри помещения и не в студии, вам подойдут фотоаппараты с внешними вспышками.
9. Если вы не ограничены в средствах, и у вас есть опыт работы с профессиональными фотоаппаратами больше года, и вы предпочитаете фотографировать внутри помещения, в студии, с использованием специальной аппаратуры, Вам подойдут такие фотоаппараты, как Nikon D70, Canon 5D.
10. Если вы не ограничены в средствах, и у вас есть опыт работы с профессиональными фотоаппаратами больше года, и вы предпочитаете фотографировать внутри помещения, в студии, без использования специальной аппаратуры, вам подойдут такие фотоаппараты, как Nikon D90, Canon D500.
11. Если вы не ограничены в средствах, и у вас есть опыт работы с профессиональными фотоаппаратами больше года, и вы предпочитаете фотографировать на улице, на природе и любите делать портреты, вам подойдут все фотоаппараты фирмы Canon или Nikon с «портретными» объективами (Canon EF 100mm f/2,8 , Nikon 50mm f/1.4G AF-S).
12. Если вы не ограничены в средствах, и у вас есть опыт работы с профессиональными фотоаппаратами больше года, и вы предпочитаете фотографировать на улице, на природе и любите снимать пейзажи, вам подойдут все фотоаппараты фирмы Canon или Nikon с широкоугольными объективами (Canon EF-S 10-22 f/3.5-4.5 USM, Nikon Af 80-200mm f/2.8 D).
13. Если вы не ограничены в средствах, и у вас есть опыт работы с профессиональными фотоаппаратами больше года, и вы предпочитаете фотографировать на улице, на природе и любите снимать всё (и портрет и пейзаж), Вам подойдут все фотоаппараты фирмы Canon или Nikon с объективами CANON EF 28-135 mm f/3.5-5.6, Nikkor 16-85mm f/3.5-5.6G ED.

Экспертная система должна на основе **этих Знаний** помочь найти фотоаппарат, наиболее подходящий вашему бюджету, опыту и предпочтениям.

1.4. Построение Машины вывода (Решателя) в виде дерева решений

На рис. 4 представлен орграф, отражающий **Знания экспертной системы**. Будем в дальнейшем каждую вершину орграфа называть "состоянием ЭС".

По существу, поиск решения экспертной системой означает "путешествие" по этому орграфу. Такое путешествие состоит из последовательности однотипных шагов, на каждом из которых пользователь должен решить, по какой дуге он пойдет из очередной вершины.

ВНИМАНИЕ! При построении дерева решений необходимо продумать, с какого вопроса начать и какими вопросами продолжить опрос пользователя, чтобы получилось компактное дерево решений.

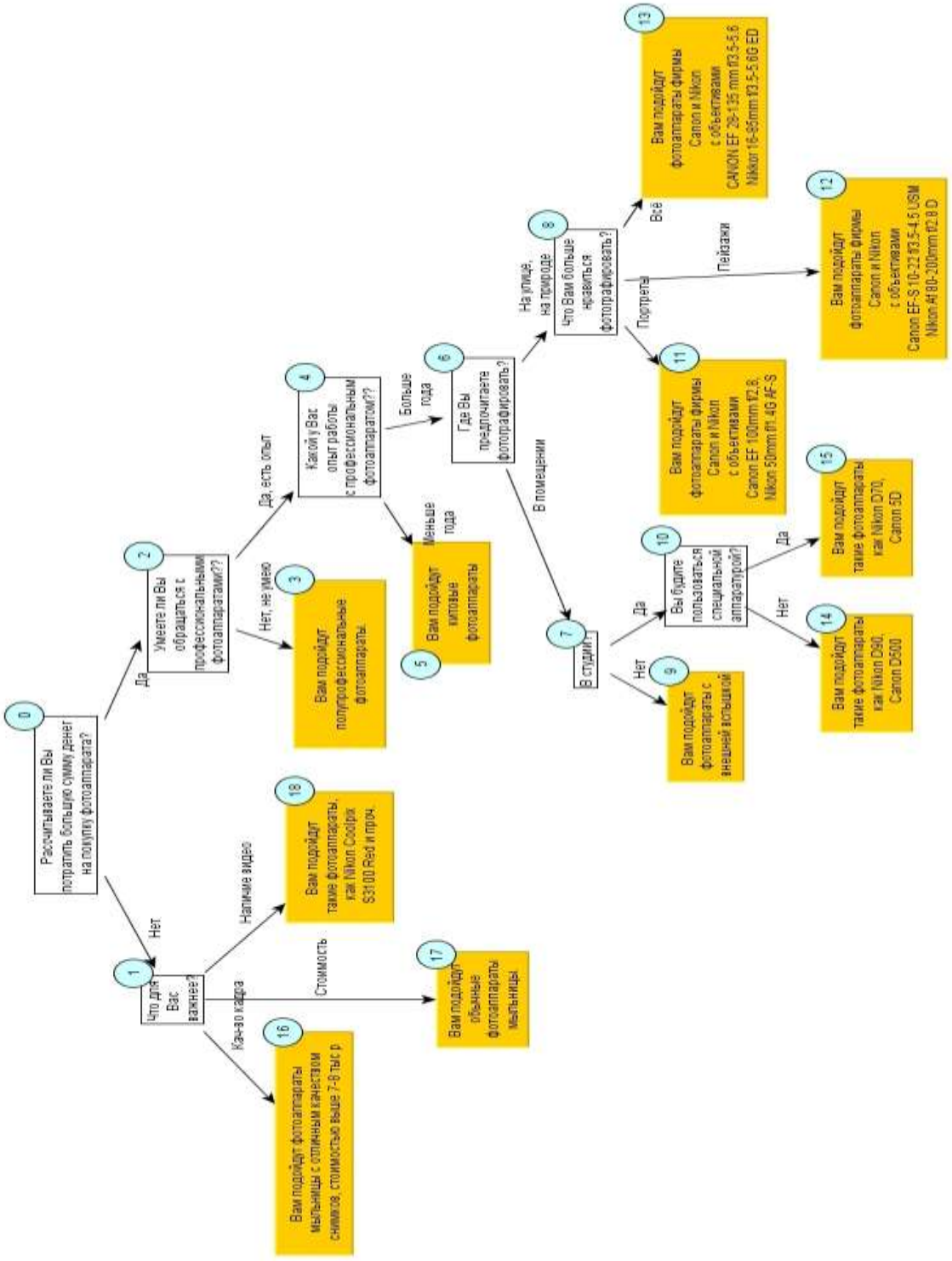


Рис 4. Дерево решений (орграф), отражающее знания ЭС

1.5. Реализации решателя с помощью таблицы переходов

Следуя реляционному подходу, мы должны теперь описать полученный орграф подходящими таблицами: каждую дугу мы опишем номером её начала и номером её конца.

Кроме двух столбцов, соответствующих началу и концу дуги, добавим еще два столбца: «Конец поиска» и «Ответ пользователя».

Назовем эту таблицу "Управление переходами состояний". Столбец «Конец поиска» будет указывать на продолжение поиска (0) или его окончание (1).

Таблица 1. Управление переходами состояний

Начальное состояние	Конечное состояние	Конец поиска	Ответ пользователя
0	1	0	Нет, я ограничен(а) в бюджете
0	2	0	Да, я не ограничен(а) в средствах
1	16	1	Качество фотографий
1	17	1	Стоимость фотоаппарата
1	18	1	Наличие видео съемки
2	3	1	Нет, не умею
2	4	0	Да, есть опыт
4	5	1	Меньше года
4	6	0	Больше года
6	7	0	В помещении
6	8	0	На улице, на природе
7	9	1	Нет, я не планирую фотографировать в студии
7	10	0	Да, я планирую фотографировать в студии
8	11	1	Портреты
8	12	1	Пейзажи
8	13	1	Всё
10	14	1	Нет, только фотоаппаратом
10	15	1	Да, хочу максимально задействовать всю технику

Таблица 1 «Управление переходами состояний» используется для реализации работы Машины вывода (Решатель). Она управляет «движением» системы от одного состояния к другому по дереву решений или выдает РЕШЕНИЕ экспертной системы и останавливается, если встречает «1» в столбце «Конец поиска».

В этом случае система должна будет выдать пользователю окончательное решение и объяснить, как было получено принятое решение (об этом подробно в Лабораторной N3).

1.6. Порядок проведения работы

1. Студент самостоятельно выбирает проблемную область (ПО) и задачу, для решения которой он будет строить ЭС.
2. Проводит онтологические исследования, составляет словарь терминов и определяет взаимосвязи объектов в выбранной ПО;
3. Строит концептуальную модель знаний – определяет важные объекты, их свойства и связи между ними;
4. Строит систему декларативных знаний.
5. Строит дерево решений (решатель) на основе имеющихся декларативных знаний.
6. Показывает, обсуждает и уточняет вместе с преподавателем на практическом занятии и на лабораторной работе концептуальную модель и построенное дерево решений.

7. Создает таблицу управления переходами состояний и программирует ее для ввода в ЭВМ.
8. Показывает и уточняет вместе с преподавателем построенную Таблицу 1 «Управление переходами состояний».

1.7. Контрольные вопросы

1. Что входит в понятие «онтологические исследования» в данной Проблемной области?
2. Что представляет собой концептуальная модель знаний?
3. Какими особенностями должна обладать концептуальная модель для представления полученных знаний деревом решений? Системой продукционных правил?
4. Какая связь между деревом решений и системой продукционных правил?
5. Как построить решатель в виде дерева решений? С чего начать?
6. Расскажите о работе решателя, представленного таблицей переходов.

Практическая работа №12
Решение прикладных задач с помощью экспертной системы реального времени
Создание интерфейса ЭС

Цель работы: построить интерфейс для вывода и ввода необходимой для работы ЭС информации (вывод вопросов ЭС к пользователю, ввод в систему ответов пользователя, вывод решений ЭС); организовать работу решателя с данными, вводимыми с интерфейса; создать компьютерную программу, реализующую диалог пользователя с экспертной системой и работу решателя на основе ответов пользователя.

1.8. Краткие общие сведения по созданию Интерфейса пользователя

Для реализации **Интерфейса пользователя** необходимо организовать диалог пользователя с ЭС. Возможен выбор ответа на экране монитора из списка предложенных ответов с использованием графического оконного интерфейса и «мышки» или ввод ответа с клавиатуры («да-нет», цифры 1, 2, …, L), которым соответствует один из альтернативных ответов.

Единственное, но очень важное требование к интерфейсу пользователя – взаимодействие с ЭС должно быть удобным для пользователя.

При выборе ответа на вопрос ЭС или после принятого ЭС решения пользователю бывают нужны разъяснения. Для поддержки и помощи пользователю в таких случаях в **Интерфейсе** необходимо предусмотреть кнопку «**?Помощь**». Нажав на неё, пользователь сможет получить развернутое объяснение от **Блока объяснений**, что имеет в виду ЭС, поставив именно этот вопрос, или как было принято экспертной системой данное решение (детали построения Блока объяснений приведены в Лабораторной N3).

Помимо кнопки «? Помощь», при построении интерфейса экспертной системы необходимо предусмотреть ещё две кнопки: кнопку «**Выход**»- выход из программы ЭС, и кнопку «**Начать сначала**»- повторить работу с ЭС.

Эти кнопки нужны для того, чтобы пользователь мог корректно завершить работу с ЭС или, если необходимо, мог повторить цикл работы с экспертной системой, не запуская программу заново.

1.9. Организация диалога пользователя с ЭС

Для того чтобы наша экспертная система могла взаимодействовать с пользователем в интерактивном режиме, одной таблицы управления переходами мало, поскольку требуется еще информация о реакциях экспертной системы на ответы пользователя. Реакция же может быть двоякой: ответ (решение) системы или очередной вопрос пользователю. Поэтому для взаимодействия пользователя с ЭС построим Таблицу 2 «Вопросы-Ответы».

В соответствии с этой таблицей и в зависимости от состояния, в котором находится наша экспертная система, на экран монитора будут выводиться вопросы к пользователю или ответы экспертной системы.

Таблица 2. Вопросы-Ответы

0	*Вопрос*: Рассчитываете ли Вы потратить большую сумму денег на покупку фотоаппарата?
1	*Ответ*: Нет, я ограничен(а) в бюджете. *Вопрос:* Что для Вас важнее?
2	*Ответ*: Да, я не ограничен(а) в средствах. *Вопрос*: Умеете ли Вы обращаться с профессиональными фотоаппаратами?
3	*Ответ:* Нет, не умею. => Вам подойдут полупрофессиональные фотоаппараты. Вопросов больше нет.
4	*Ответ:* Да, есть опыт. *Вопрос:* Какой у Вас опыт работы с профессиональным фотоаппаратом?
5	*Ответ:* Меньше года. => Вам подойдут китовые фотоаппараты. Вопросов больше нет.
6	*Ответ:* Больше года. *Вопрос*: Где Вы предпочитаете фотографировать?
7	*Ответ:* В помещении. *Вопрос*: В студии?
8	*Ответ*: На улице, на природе. *Вопрос*: Что Вам больше нравиться фотографировать?
9	*Ответ*: Нет, я не планирую фотографировать в студии. =>Вам подойдут фотоаппараты с внешней вспышкой. Вопросов больше нет.
10	*Ответ*: Да, я планирую фотографировать в студии. *Вопрос*: Вы будите пользоваться специальной аппаратурой?
11	*Ответ:* Портреты => Вам подойдут такие фотоаппараты фирмы Canon и Nikon с объективами Canon EF 100mm f/2,8 , Nikon 50mm f/1.4G AF-S. Вопросов больше нет.
12	*Ответ:* Пейзажи. => Вам подойдут фотоаппараты фирмы Canon и Nikon с объективами Canon EF-S 10-22 f/3.5-4.5 USM, Nikon Af 80-200mm f/2.8 D. Вопросов больше нет.
13	*Ответ:* Люблю все фотографировать. => Вам подойдут фотоаппараты фирмы Canon и Nikon с объективами CANON EF 28-135 mm f/3.5-5.6, Nikkor 16-85mm f/3.5-5.6G ED. Вопросов больше нет.
14	*Ответ*: Нет, только фотоаппаратом. => Вам подойдут такие фотоаппараты как Nikon D90, Canon D500. Вопросов больше нет.
15	*Ответ:* Да, хочу максимально задействовать всю технику. => Вам подойдут такие фотоаппараты как Nikon D70, Canon 5D. Вопросов больше нет.

16	*Ответ:* Качество изображений. => Вам подойдут фотоаппараты мыльницы с отличным качеством снимков, стоимостью выше 7-8 тыс. руб. Вопросов больше нет.
17	*Ответ:* Стоимость. => Вам подойдут обычные фотоаппараты мыльницы. Вопросов больше нет.
18	*Ответ:* Наличие видеосъемки. => Вам подойдут такие фотоаппараты, как Nikon Coolpix S3100 Red и проч. Вопросов больше нет.

1.10. Реализация работы решателя в соответствии с ответами пользователя

Для того чтобы реализовать работу решателя в соответствии с ответами пользователя, необходимо организовать взаимодействие Таблицы 1 «Управление переходами состояний» и Таблицы 2 «Вопросы-ответы».

Для работы экспертной системы нам в каждый момент потребуется знать, в каком состоянии находится система.

Создадим управляющую ячейку (Таблица 3), в которую будем заносить номер текущего состояния системы. Эта ячейка поможет нам реализовать работу Решателя в соответствии с ответами пользователя.

Таблица 3. Текущее состояние

Текущее состояние
0

В начальном состоянии Экспертной системы значение в ячейке равно нулю. Ясно, что должно происходить дальше.

➔ Из **Таблицы 2 «Вопросы и ответы»** на экран ЭВМ выводится ВОПРОС, соответствующий текущему состоянию системы.

Пользователь вводит ОТВЕТ (с клавиатуры или мышкой), который сравнивается со всеми возможными для текущего состояния ответами в **Таблице 1 «Управление переходами состояний»** (сравнивают с ответами, находящимися в столбце «ответ пользователя»).

Выбирают ту строку Таблицы 1, которой соответствует ответ, выбранный пользователем, и в управляющую ячейку (**Таблица 3**) вносят новое значение «текущего состояния», которое находится в найденной строке в столбце «конечное состояние».

Затем проверяется запись в ячейке «конец поиска». Если поиск не закончен (в ячейке «конец поиска» записан ноль), то цикл «вопрос ЭС - ответ пользователя» продолжается с пункта, обозначенного стрелкой ➔.

Если в ячейке «конец поиска» записана единица («1» = да, конец поиска), то Экспертная Система выдает на экран РЕШЕНИЕ, соответствующее «текущему состоянию» из **Таблицы 2 «Вопросы- Ответы»,** переходит к **Блоку объяснений** (об этом будет сказано дальше) и останавливается.

1.11. Порядок проведения Лабораторной работы № 2

1. Студент выбирает любой способ организации диалога с пользователем (графический оконный интерфейс и «мышку» для выбора возможных ответов, вв3од ответа с клавиатуры в соответствии с одним из возможных альтернативных ответов, и т.д.).

2. Предусматривает в интерфейсе кнопку «**? Помощь**» для поддержки пользователя при выборе им ответа на запрос ЭС и объяснения принятых ЭС решений, а также кнопку «**Выход**» для выхода из программы, и кнопку «**Начать сначала**» для повторения цикла работы с ЭС.
3. Студент строит Таблицу 2 «Вопросы-Ответы» и Таблицу 3 «текущее состояние» системы.
4. Создает программу, которая организует вывод на интерфейс вопросов и ответов из Таблицы 2 «Вопросы-Ответы».
5. Создает программу, обеспечивающую цикл перехода системы в новое состояние в зависимости от ответа пользователя, и выдачу нового вопроса пользователю или ответа ЭС в зависимости от текущего состояния системы (реализация взаимодействия Таблицы 1, Таблицы 2 и Таблицы 3).
6. Отлаживает программу, реализующую интерфейс пользователя и взаимодействие пользователя с ЭС.
7. Показывает и вместе с преподавателем уточняет интерфейс пользователя.

Контрольные вопросы к

1. Как можно организовать интерфейс пользователя?
2. Что необходимо учесть при построении интерфейса?
3. Каким требованиям должен удовлетворять интерфейс пользователя?
4. Как организовать работу Экспертной системы, чтобы учесть ответы пользователя на вопросы ЭС?
5. Что значит «удобный» интерфейс пользователя?

Практическая работа №13

Работа со стандартами разработки программного обеспечения

1.12. Краткие общие сведения по созданию Блока объяснений

Непременным элементом экспертной системы считается **Блок объяснений**. Он должен разъяснять пользователю, почему экспертная система поступает так, а не иначе. Мы предлагаем реализовать его с помощью одной кнопки «**?Помощь**» или нескольких кнопок («**Объяснение вопроса**», «**Объяснение решения**»), которые пользователь будет нажимать в зависимости от того, что именно ему непонятно. В ответ система будет разъяснять ему соответствующий аспект текущего состояния.

Давайте начнем с кнопки «**Объяснение вопроса**», которую пользователь будет нажимать, если ему непонятен вопрос. Объяснение вопросов должно содержать справочную информацию, облегчающую правильный выбор ответа.

Кнопка «**Объяснение решения**» должно указывать путь получения решения (цепочка выбранных ответов), который привел к полученному решению. Если это необходимо, то дополнительно может быть дана информационно-аналитическая справка о полученном решении или об интерпретации решения, и приведены необходимые иллюстрации.

1.13. Создание Блока объяснений

Все эти развернутые формулировки с объяснениями будут храниться в одной таблице.

Для реализации **Блока объяснений** необходимо создать **Таблицу 4** «**Объяснения**», соответствующую в общем случае **всем возможным состояниям ЭС**. В более частном

случае, если вопросы к пользователю не требуют разъяснений, то в таблице Объяснения будут указаны только те состояния, которые соответствуют решениям ЭС.

В первом столбце таблицы нужно указать текущее состояние системы, в соседнем столбце - необходимые пользователю объяснения.

Когда пользователь нажимает на клавишу «**?Помощь**» на интерфейсе пользователя на экран дисплея из Таблицы 4 должен быть выведен текст объяснения, соответствующий текущему состоянию системы. Чтобы закрыть форму с объяснениями, нужно щелкнуть по крестику в верхнем углу формы.

За основу Таблицы 4 «Объяснения» можно взять Таблицу 1 «Управление переходами состояний» и Таблицу 2 «Вопросы-ответы». Для терминальных (конечных) состояний, когда у системы уже нет вопросов, система будет сообщать пользователю, как был получен ответ.

В итоге таблица «Объяснения» будет выглядеть примерно так:

Таблица 4 «Объяснения»

Состояние	Текст
0	От Вашего бюджета зависит многое. Собираетесь ли Вы потратить, например, 5 тысяч или 30 тыс.?
1	Вы выбрали вариант ответа "Нет, я ограничен(а) в бюджете", поэтому мы предлагаем вам простые фотоаппарата-мыльницы. В настоящее время уровень качества изображения в таких фотоаппаратах стал намного выше.
2	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах". Вы рассчитываете потратить приличную сумму на покупку фотоаппарата, поэтому лучше всего покупать профессиональный фотоаппарат. У него качество изображения намного выше.
3	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Нет, не умею пользоваться профессиональными фотоаппаратами". Поэтому предлагаем купить полупрофессиональные фотоаппараты - у них качество изображения лучше и они просты в использовании.
4	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов". Вы умеете обращаться с профессиональным фотоаппаратом. Но если Вы только новичок - дорогую технику покупать не стоит.
5	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов", с опытом работы "Меньше года". Поэтому советуем Вам приобрести китовые фотоаппараты - это профессиональные фотоаппараты со стандартным объективом.
6	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов", с опытом работы "Больше года". Поэтому советуем Вам приобрести фотоаппараты с объективом, отличным от обычного. Это позволит Вам осуществить Ваши творческие задумки и идеи.
7	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов", с опытом работы "Больше года", предпочитаете фотографировать "Внутри помещения".

8	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов", с опытом работы "Больше года", предпочитаете фотографировать "На улице, на природе".
9	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов", с опытом работы "Больше года", предпочитаете фотографировать "Внутри помещения". Далее Вы выбрали вариант помещения "Не в студии". Поэтому Вам подойдут фотоаппараты с внешними вспышками.
10	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов", с опытом работы "Больше года", предпочитаете фотографировать "Внутри помещения". Далее Вы выбрали вариант помещения "Студия".
11	Вы выбрали вариант ответа "Да, я не ограничен (а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов", с опытом работы "Больше года", предпочитаете фотографировать "На улице, на природе". Далее Вы выбрали жанр съемки "Портрет". Поэтому Вам подойдут все фотоаппараты фирмы Canon или Nikon с объективами "портретник"
12	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов", с опытом работы "Больше года", предпочитаете фотографировать "На улице, на природе". Далее Вы выбрали жанр съемки "Пейзаж". Поэтому Вам подойдут все фотоаппараты фирмы Canon или Nikon с широкоформатными объективами.
13	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов", с опытом работы "Больше года", предпочитаете фотографировать "На улице, на природе". Далее Вы выбрали жанр съемки "Все". Поэтому Вам подойдут все фотоаппараты фирмы Canon или Nikon с объективами CANON EF 28-135 mm f/3.5-5.6, Nikkor 16-85mm f/3.5-5.6G ED."
14	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов", с опытом работы "Больше года", предпочитаете фотографировать "Внутри помещения". Далее Вы выбрали вариант помещения "Студия" "Без использования специальной аппаратуры". Поэтому рекомендуем Вам самые качественные фотоаппараты фирмы Canon и Nikon.
15	Вы выбрали вариант ответа "Да, я не ограничен(а) в средствах", а после - "Да, есть опыт в использовании профессиональных фотоаппаратов", с опытом работы "Больше года", предпочитаете фотографировать "Внутри помещения". Далее Вы выбрали вариант помещения "Студия" "С использованием специальной аппаратуры". Поэтому Вам подойдут такие фотоаппараты как Nikon D70, Canon 5D.
16	Вы выбрали вариант ответа "Нет, я ограничен(а) в бюджете", и для Вас важнее "Качество изображений". Поэтому Вам стоит приобрести фотоаппараты стоимостью от 7 тыс. руб., имеющие от 10 мегапикселей качество изображения
17	Вы выбрали вариант ответа "Нет, я ограничен(а) в бюджете", и для Вас важнее "Стоимость фотоаппарата". Поэтому можете купить любой фотоаппарат, имеющийся в магазине по любой цене.
18	Вы выбрали вариант ответа "Нет, я ограничен(а) в бюджете", и для Вас важнее "Наличие видеосъемки". Поэтому попросить продавца-консультанта показать Вам все модели фотоаппаратов, поддерживающих видеосъемку

Для многих проблемных областей очень желательно (и даже необходимо) помимо справочного или разъясняющего текста использовать иллюстрации, в которые включены объекты, раскрывающие смысл текста (изображения объектов, музыка, видео, другое).

1.14. Тестирование и отладка созданной Экспертной Системы

Созданная Экспертная Система должна пройти всестороннюю проверку.
Особое внимание нужно обратить на следующие важные компоненты ЭС:

1. Интерфейс пользователя: насколько он удобен для самого пользователя.
2. Вопросы и ответы: с какого вопроса начать и какими вопросами продолжить опрос пользователя, чтобы получилось компактное дерево решений.
3. Блок объяснений: ЭС должна быть «дружественной». Это значит, она должна поддерживать пользователя при всех вопросах, ответах и решениях, вызывающих у него затруднения. Сообщение типа: решения нет – недопустимо.
4. Ошибки: грамматические, синтаксические, ошибочные решения, неправильное объяснение результатов и т.д. в работе ЭС недопустимы.

Все недоработки, неточности, ошибки должны быть найдены и исправлены в процессе тестирования ЭС.

1.15. Порядок проведения

1. Студент записывает все шаги (ответы пользователя), которые ведут к решению, предлагаемому ЭС, подбирает разъясняющий текст, справочные материалы, иллюстрации и строит Таблицу 4 «Объяснения».
2. Он создает программу, реализующую Блок объяснений: при нажатии на клавишу «**? Помощь**» на экран дисплея будет выводиться специальное «окно» с необходимыми пользователю разъяснениями и иллюстрациями.
3. Студент тестирует и отлаживает работу программы, реализующей Экспертную систему в полном объеме: интерфейс пользователя, решения, принимаемые экспертной системой, работу блока объяснений.
4. Проводит опытную эксплуатацию ЭС совместно с преподавателем. Если это необходимо, он исправляет выявленные неполадки и повторяет тестирование работы ЭС вместе с преподавателем.
5. Студент пишет информационно-аналитическую справку (отчёт) о построенной ЭС (требования к отчету даны в **Приложении А**) и защищает выполненную работу в ходе устной беседы с преподавателем.
   Во время беседы проверяется умение студентов отвечать на контрольные вопросы по теоретической и практической части работы.
6. По результатам лабораторных работ студент получает зачёт.

1.16. Контрольные вопросы

1. Что такое экспертная система?
2. Основные компоненты ЭС?
3. Рассказать об этапах разработки ЭС.
4. Рассказать о компонентах блока объяснений экспертной системы.
5. Зачем нужно тестирование и опытная эксплуатация ЭС?

6. В чём может заключаться доработка ЭС?

Практическая работа №14
Анализ отчетной документации различных этапов разработки информационных систем

1. Исторические этапы развития информационных систем.
2. Основные методологические подходы анализа и проектирования: принципы и средства.

1. Исторические этапы развития информационных систем

Научно-техническая революция, широко развернувшаяся во второй половине XX века, породила надежды на то, что с помощью новых научных дисциплин и новой техники будут разрешены трудные проблемы и противоречия человеческой жизни.

Цель создания компьютера, как и любого нового технического средства, определялась необходимостью появления нового инструмента, позволяющего ускорить процесс вычислений. И только через 20 лет после его изобретения компьютер стали рассматривать в качестве универсального средства обработки информации. В последнее время он выступает не только средством эффективного управления, но и базовым инструментом новой информационной технологии.

Сегодня информационные технологии оказывают влияние как на обработку данных, так и на способ выполнения работы людьми, на продукцию, характер конкуренции. Информация становится ключевым ресурсом организаций, а информационная обработка – делом стратегической важности. Большинство организаций не сможет успешно конкурировать, пока не предложит своим клиентам такой уровень обслуживания, который возможен лишь при помощи систем, основанных на высоких технологиях.

Можно выделить, по крайней мере, три причины необходимости автоматизации информационных процессов.

1. Второй информационный барьер.

В середине 70-х годов человечество подошло в своем развитии к так называемому второму информационному барьеру, то есть достигло такого состояния, когда потоки и количество информации настолько возросли, что люди, населяющие планету, не могли переработать всю циркулирующую в обществе информацию.

2. Противоречие между своевременностью и достоверностью информации.

Допустим, что нам надо управлять некоторым объектом, который в момент времени t0 находится в состоянии s0. Для выработки управляющего воздействия необходимо собрать и иметь об объекте информацию i0. Однако в течение времени сбора информации к моменту t1 объект перейдет в состояние s1, то есть информация о нем i0 будет уже недостоверна. Это противоречие усилится, если объект управления имеет многоуровневую структуру.

3. Необходимость интеграции общих информационных источников для принятия решений при эффективном управлении.

Современная экономика немыслима без эффективного управления. Успех управления во многом определяется эффективностью принятия интегрированных решений, которые учитывают самые разносторонние факторы и тенденции динамики их развития.

Обобщая вышесказанное, отметим, что одной из причин активного развития данной области является то, что автоматизация в первую очередь служит основой коренного изменения процессов управления, играющих важную роль в деятельности человека и общества. При этом данный процесс является на сегодня одной из самых ресурсоемких сфер деятельности техногенного общества.

Обратимся к истории вопроса развития и становления информационных систем. Известно, что каждое новое десятилетие развития вычислительной техники определяло не только новый вид компьютера, но и несло с собой новую технологию обработки информации. Последнее предопределяло, с одной стороны, переход информационных систем на новый уровень автоматизации, а с другой – новый этап внедрения автоматизированных систем.

Представим краткую характеристику каждого из таких этапов (такое разделение весьма условно и определяется, как было отмечено выше, определенным периодом развития вычислительной техники).

Этап 1. 50-е годы. Это время характеризуется появлением новой технологии – языки программирования. В эти годы информационные системы использовались для обработки счетов и расчета зарплаты, а реализовывались на электромеханических бухгалтерских счетных машинах. Это приводило к некоторому сокращению затрат и времени на подготовку бумажных документов. Такие системы называются системами обработки транзакций. К транзакциям относят следующие операции: выписка счетов, накладных, составление платежных ведомостей и другие операции бухгалтерского учета.

Этап 2. В 60-е годы средства вычислительной техники получили дальнейшее развитие: появляются операционные системы, дисковая технология, совершенствуются языки программирования. Развитие вычислительной техники обусловило появление новых возможностей в автоматизации различных видов деятельности, например, подготовки отчетной документации. ЭВМ теперь используются не только для вычислений, но и как средство накопления и обработки информации. Другими словами, этот период можно определить как начальный этап автоматизации информационных систем.

Данный этап характеризуется следующими факторами:

- накопление компаниями первоначального опыта использования ЭВМ, выявление основных направлений применения;
- сокращение управленческих расходов и численности персонала;
- автоматизация отдельных операций (бухгалтерский учет, финансовые расчеты, материально-техническое снабжение);
- появление систем управленческих отчетов (СУО), ориентированных на менеджеров, принимающих решения.

Этап 3. Достижения в области вычислительной техники в период с 1968 по 1972 год (базы данных, терминалы, сети и постоянные запоминающие устройства) привели к тому, что почти все существовавшие до тех пор приложения устарели, и возникла потребность в целом классе новых приложений. Концепцией интерактивных баз данных заинтересовались менеджеры, которые предложили ряд идей по устранению информационной изоляции:

- рассматривать информацию как ключевой корпоративный ресурс;
- построить согласованную, интегрированную базу данных, охватывающую все сферы деятельности компании;
- обеспечить ввод и обновление информации в базу данных немедленно, как только она поступает в компанию.

Главным результатом 70-х годов было то, что сформировалось видение будущего: информация всегда доступна, оперативна, организована в согласованной базе данных. Информация стала рассматриваться как стратегическое сырье наравне с другими ресурсами. Этот этап внедрения автоматизированных систем – этап установления контроля над внедрением новой информационной технологии – характеризовался следующим:

- поиск сфер применения ЭВМ в управлении завершен, парк ЭВМ не расширяется;
- вырабатываются организационные формы управления новой техникой, выявляется ее влияние на управленческие процессы в целом;
- отдельные виды информационных систем компании пока изолированы и слабо совместимы между собой;
- информационные системы предназначены в основном для информационного обеспечения руководства фирмы;
- выдвигается идея интеграции информационного обеспечения управленческих процессов, в компаниях создаются единые, самостоятельные информационные службы.

Кроме того, в 70-х – начале 80-х гг. в офисах начинают применять разнообразные компьютерные и телекоммуникационные технологии, которые расширили область применения информационных систем. К таким технологиям относятся: текстовая обработка, настольное издательство, электронная почта и др. При этом информационные системы широко используются в качестве средства управленческого контроля, поддерживающего и ускоряющего процесс принятия решений.

Этап 4. 80-е гг. В это время компьютерная индустрия вступила в пору зрелости. Этот период стал периодом перехода процесса разработки приложений для бизнеса из области искусства в инженерную дисциплину. В ходе этого процесса сформировались два важных результата: методология и CASE-технология (Computer Aided Software Engineering). При этом методология рассматривалась как формальное предписание, руководство, описывающее процесс построения программных средств, тогда как CASE – как сам процесс проектирования. В ходе построения методологии были созданы программные инструментальные средства, которые поддерживали и методологию, и сам процесс проектирования – CASE.

В эти годы проходил этап интеграции информационных систем и видов деятельности. Его характерные особенности:

- преодолены технические трудности, вычислительные центры и сети объединяются с системами другой природы – телефоном, телевидением и т.д.;
- перенесен акцент автоматизации на создание децентрализованных систем, в которых все ПК, ЭВМ, разнородное оборудование фирмы объединены в локальную сеть, в таких системах информация распределена между ПК, а управленец приближен к компьютеру;
- преодолены организационно-методологические трудности, выдвигается концепция управления информационными ресурсами, в которой информация рассматривается как еще один важный организационный ресурс наряду с финансами, материалами, оборудованием, персоналом и т.д.;
- преодолены психологически трудности, сформирован новый эшелон работников, которые рассматривают новую технику как инструмент современного производства;
- происходит интеграция информационных систем на единой технической, организационной, методологической основе;
- повышение статуса информационной службы, преобразованной в службу управления информационными ресурсами предприятия.

Этап 5. В 90-е годы технические результаты технологии персональных компьютеров стали очевидны: легкий в применении графический интерфейс пользователя, фактически неограниченный доступ к ресурсам компьютера с индивидуального рабочего места и новое представление о компьютерах и приложениях, выполняемых на них. Появляются новые технологии обработки данных – распределенная обработка данных.

Начало нового тысячелетия характеризуется появлением огромного числа новых информационных технологий, к которым можно отнести геоинформационные технологии, Intranet-технологию, OLAP-технологию и другие. По мнению ведущих ученых, в настоящее время начался переход развитых стран из века энергетики в век информации, о чем свидетельствуют следующие симптомы:

- время удвоения накопленных научных данных составляет 2-3 года;
- материальные затраты на хранение, передачу и переработку информации превышают аналогичные расходы на энергетику.

Развитие Российской IT-индустрии, безусловно, находится в прямой зависимости от развития мировой IT-индустрии. При этом развивающийся процесс информатизации в России имеет свои национальные особенности, которые на сегодняшний день можно охарактеризовать в целом следующим образом:

- локальная фрагментарная автоматизация;
- практика человеко-проект;
- зависимость от конкретного исполнителя;
- разрыв уровня задач и информационных технологий;
- отсутствие исторически накопленных данных;
- финансирование по остаточному признаку.

Однако за последние 10 лет Российская IT-индустрия сильно изменилась. Об этом говорит и участие первых лиц государства в обсуждении вопросов информатизации на международных форумах, появление программы «Электронная Россия», участие Российской IT-индустрии в крупных проектах не только в России, но и за ее пределами.

Подведем некоторые итоги. Ниже перечислены ключевые моменты по каждому из представленных этапов развития автоматизированной обработки информации.

1-й этап. Первые информационные системы появились в 50-х гг. В эти годы они были предназначены для обработки счетов и расчета зарплаты, а реализовывались на электромеханических бухгалтерских счетных машинах. Это приводило к некоторому сокращению затрат и времени на подготовку бумажных документов.

2-й этап. 60-е гг. знаменуются изменением отношения к информационным системам. Информация, полученная из них, стала применяться для периодической отчетности по многим параметрам. Для этого организациям требовалось компьютерное оборудование широкого назначения, способное обслуживать множество функций, а не только обрабатывать счета и считать зарплату, как было ранее.

3-й этап. В 70-х – начале 80-х гг. информационные системы начинают широко использоваться в качестве средства управленческого контроля, поддерживающего и ускоряющего процесс принятия решений.

4-й этап. К концу 80-х гг. концепция использования информационных систем вновь изменяется. Они становятся стратегическим источником информации и используются на всех уровнях организации любого профиля (это характерно и для сегодняшнего дня). Информационные системы этого периода, предоставляя вовремя нужную информацию, помогают организации достичь успеха в своей деятельности, создавать новые товары и услуги, находить новые рынки сбыта, обеспечивать себе достойных партнеров, организовывать выпуск продукции по низкой цене и многое другое.

Таким образом, каждый из этапов развития автоматизированной обработки информации характеризуется следующими позициями: 1) целью использования информационных систем; 2) технической основой; 3) технологией; 4) концепцией использования информации в организации. Кроме того, применяемые в тот или иной период информационные системы обладают характерными особенностями. Кратко данную информацию можно представить в виде таблицы (табл. 1).

Этапы автоматизации обработки информации

Этап	**Цель использования**	**Техническая основа**	**Технология**	**Концепция использования информации**	**Особенности применяемых информационных систем**
50-е гг.	Повышение скорости обработки документов Упрощение процедуры обработки счетов и	Электронная вычислительная машина	Языки программирования	Бумажный поток расчетных документов	Информационные системы обработки расчетных документов на электромеханических

	расчета зарплаты				бухгалтерских машинах
60-е гг.					
70-е гг.					
80-е гг.					
90-е гг.					

Задания для самостоятельной работы

1. Заполните таблицу 1.
2. Составьте четыре вопроса, позволяющих осветить представленный материал.
3. Дополните представленную информацию сведениями по современным перспективам развития ИС в XXI веке.
4. Укажите новые технологии, которые сегодня используются для функционирования ИС (подготовить ответ на лекцию).
5. Каковы современные области применения и примеры реализации ИС? (подготовить ответ на лекцию)
6. На основе представленной информации, докажите, что на современном этапе развития экономики принятие интегрированных решений невозможно без автоматизированной системы обработки (данных) информации.

2. Основные методологические подходы анализа и проектирования: принципы и средства

Методологию можно определить как совокупность взглядов на то, какой должна быть последовательность шагов и какова их взаимосвязь при разработке программного обеспечения.

Методология реализуется через конкретные технологии и поддерживающие их стандарты, методики и инструментальные средства, которые обеспечивают выполнение процессов ЖЦ.

Технологии проектирования - инструментальные средства, поддерживающие сам процесс проектирования.

В настоящее время можно выделить **три основных методологических подхода к анализу и проектированию:**

- структурный подход;
- объектно-ориентированный подход;
- информационная инженерия.

Структурный подход – метод исследования системы, изучение которой начинается с ее общего обзора, последующей детализации, созданием иерархической структуры с достаточным числом уровней.

Объектно-ориентированный подход – метод анализа предметной области, основанный на выявлении объектов и установлении взаимных связей между ними.

Информационная инженерия – это метод исследования системы, который базируется на следующих позициях: информационный анализ предметных областей (бизнес - областей); информационное моделирование – построение комплекса взаимосвязанных моделей данных; системное проектирование функций обработки данных; детальное конструирование процедур обработки данных.

Сравнительная характеристика СА и ООП.

Рассмотрим в сравнении получившие наибольшее распространение подходы структурного и объектно-ориентированного анализа, определив для каждого из них наиболее известные методологии, принципы, их характеризующие, средства и технологии (табл. 2).

Таблица 2
Сравнительная характеристика СА и ООП

Структурный подход	**ОО подход**
Определение	
Метод исследования системы, изучение которой начинается с ее общего обзора, последующей детализации, созданием иерархической структуры с достаточным числом уровней	Метод анализа предметной области, основанный на выявлении объектов и установлении взаимных связей между ними
Принципы	
Основные: **принцип "разделяй и властвуй"** – принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения; **принцип иерархического упорядочивания**– принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.	1. Абстрагирование. 2. Инкапсуляция. 3. Модульность. 4. Иерархия (иерархическая организация). **Основные абстракции: Объект** – абстракция некоторой сущности предметной области (объект реального мира) или программной системы (архитектурный объект), обладающая **состоянием** (state),**поведением** (behavior) и**индивидуальностью** (identity). **Класс** – множество объектов, разделяющих общие свойства, поведение, отношения и семантику. Класс инкапсулирует (объединяет) в себе данные (атрибуты) и поведение (операции).

Кроме того, учитываются и **другие принципы**: 1. принцип абстрагирования (выделение существенных аспектов системы и отвлечения от несущественных); 2. принцип формализации (применение строгого методического подхода к решению проблемы); 3. принцип непротиворечивости (обоснованность и согласованность элементов); 4. принцип структурирования данных (данные должны быть структурированы и иерархически организованы).	Объект - экземпляр (instance) класса
Методологии	
Наибольшее распространение получили следующие методологии: • SADT (Structured Analysis and Design Technique) методология; (стандарт) • методология Гейна Сарсона; • методология Йодана/Де Марко; • методология информационного моделирования Мартина; • методология развития структурных систем Варнье-Орра; • методология развития систем Джексона и др.	В качестве методологии выступает унифицированный язык моделирования **UML, который представляет собой синтез трех методов** 1. Метод Booch, созданный Грейди Бучем (нашел применение на этапах проектирования и разработки различных программных систем). 2. OOSE (Object-Oriented Software Engineering) - Айвар Джекобсон (Ivar Jacobson) (содержал средства представления вариантов использования, которые имеют существенное значение на этапе анализа требований в процессе проектирования бизнес-приложений.). 3. OMT (Object Modeling Technique) – Джеймс Рамбо (James Rumbaugh) (особенно полезен для анализа и разработки.
Средства	
1. Диаграммы, иллюстрирующие функции, которые должна выполнять система, и связи между этими функциями – для этой цели чаще всего используются **DFD** (Data Flow	1. Диаграммы, определяющие статическую структуру приложения (диаграммы классов, объектов, компонентов и диаграмма развертывания);

Diagrams) и **SADT(IDEF0)** (Structured Analysis and Design Technique. 2. Диаграммы, **моделирующие данные** и их взаимосвязи – **ERD** (Entity-Relationship Diagrams) 3. Диаграммы, **моделирующие поведение**системы – **STD** (State Transition Diagrams)	2. Диаграммы, представляющие различные аспекты динамического поведения (диаграммы прецедентов, последовательности, деятельности, кооперации и состояний); 3. Диаграммы, определяющие способы организации и управления программными модулями (диаграммы пакетов, подсистем и моделей)

Задания для самостоятельнойработы

1. Изучите текст, ответьте на следующие вопросы.
2. В чем суть структурного подхода к анализу и проектированию?
3. В чем суть объектно-ориентированного подхода к анализу и проектированию?
4. В чем суть информационной инженерии как методологического подхода к анализу и проектированию?
5. На какие принципы опирается каждый из подходов? Дайте краткую им характеристику.
6. Назовите и охарактеризуйте каждый тип диаграмм структурного подхода к анализу и проектированию.
7. Назовите и охарактеризуйте каждый тип диаграмм объектно-ориентированного подхода к анализу и проектированию.
8. Дополните таблицу, вписав ключевые положения каждой из обозначенных методологий каждого из подходов.

Практическая работа №15

Создание простейших HTML-страниц в текстовом редакторе

Представление о языке HTML. Тэги. Атрибуты.

Web-страницы обычно создаются на языке HTML. HTML – hyper text markup language, что переводится как «Язык разметки гипертекста». Этот язык представляет собой набор тегов для управления разметкой документа. Файлы Web-страниц имеют расширение htm, html. На самом деле это обычные текстовые файлы, без форматирования, то есть их можно создавать, используя обычный текстовый редактор «Блокнот». Эти файлы содержат текст Web-страницы, а так же теги, которые управляют оформлением страницы и вставкой рисунков в файл.

Тег - это инструкция для браузера, как он должен отображать Web-страницу.

Теги могут быть парными и не парными.

Парные теги:

<HTML></HTML>; где <HTML> - открывающий тег; </HTML> - закрывающий тег;

<BODY></BODY>; где <BODY> - открывающий тег; </BODY> - закрывающий тег.

Между открывающим и закрывающим тегом находится тело тега. Парные теги действуют только на часть документа, ограниченную тегом.

Непарные теги:

; <HR> - для их использования не требуется второго (закрывающего) тега.

Непарные теги имеют разовое действие, или действуют на весь документ.

Многие теги могут иметь атрибуты (параметры).

Пример:

<FONT *COLOR=RED SIZE=5*>,
где COLOR и SIZE – атрибуты тега FONT, aRED и 5 - это параметры тегов.
Атрибуты управляют отображением элементов Web-страницы, а параметры тегов устанавливают значения этих атрибутов. Атрибуты отделяются друг от друга и от тега пробелами. Параметр отделяется от атрибута знаком равенства.
При отображении документа в браузере сами теги не отображаются, но влияют на способ отображения документа.

2. Структура web - страницы

Ниже приведены основные теги при написании Web-страницы. Именно с них и начинается создание страницы.

```
<HTML>
    <HEAD>
        <TITLE> Заголовок </TITLE>
    </HEAD>
    <BODY>
        Тело  веб- страницы
    </BODY>
</HTML>
```

Тег HTML открывает и закрывает страницу. Тег парный.

Тег HEAD определяет заголовок страницы. В нем находится тег TITLE и некоторые другие служебные теги. Тег парный.

Тег TITLE определяет текст строки заголовка, который будет отображаться в браузере при просмотре Web-страницы. Для удобства чтения и редактирования тегов HTML рекомендуется вложенные теги смещать относительно основного на 2-3 пробела вправо, как показано в структуре. И ещё желательно между тегами и вложенным текстом делать пробел, тогда можно будет всегда использовать способ перемещения по тексту на слово вправо-влево (ctrl + right/left).

Внутри тега <Body> - заключается тело Web-страницы, то есть находится весь текст и рисунки, расположенные на странице. Тег парный.

Атрибуты тега Body:

link - определяет цвет гиперссылок;
text - определяет цвет текста на странице;
background - определяет файл фонового рисунка;
bgcolor - определяет цвет фона страницы.

Пример:

```
<BODY BACKGROUND="al.jpg">
```

3. Определение цвета в HTML

3.1. Цифровой способ

Цвет определяется тремя парами цифр в шестнадцатеричной системе счисления. Например: 0029ff.

Соответствие между десятеричной и шестнадцатеричной системой счисления:

Дес.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Шест.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f

Каждая пара цифр определяет интенсивность одного из трех цветов модели RGB (red - красный, green - зеленый, blue - синий).

Например: 000000 - чёрный цвет, ffffff - белый цвет, ff0000 - красный цвет, 00ff00 -

зеленый цвет, 0000ff - синий цвет. Основные цвета, перемешиваясь в одинаковых пропорциях образуют белый или серый цвет, например, это различные оттенки серого: 444444, 999999, аааааа.

3.2. Определение цвета константой

Существует около двухсот констант для определения цвета. Это названия цветов в английском языке. Вот некоторые из них: red, blue, green, black, white, gold, orange, yellow, coral, aqua, cyan, gray, violet.

Пример

Определение цвета фона и текста на странице в теге Body. <BODYbgcolor=000000 text=red>

В данном примере цвет фона страницы - чёрный, а цвет текста на странице - красный.

4. Комментарии на страницах

Комментарии в HTML заключаются в тег <! ------ >.

Пример

<! -- Это комментарий и он не будет отображаться на странице -->

Комментарии при просмотре web-страницы не отображаются. Они нужны разработчику при редактировании страницы.

5. Имена web-страниц

Главная страница сайта обязательно должна называться "index". В зависимости от того на каком языке она написана, к этому имени будут добавляться соответствующие расширения - index.html, index.php, и т.д.

Индексный файл (это файл главной страницы сайта) ОБЯЗАТЕЛЬНО должен находиться в корневой папке сайта.

1. Имя любой страницы НЕ должно содержать заглавных букв, то есть должно быть написано маленькими буквами!

2. Имена любых страниц сайта НЕ должны содержать русских (национальных) букв, допустима только латиница!

Например, нельзя называть страницу заказ.html, или Zakaz.html, правильно будет так: zakaz.html.

Цель: сформировать навыки создания информационной системы на стадии предпроектного обследования, проектирования и разработки. Изучение языка разметки гипертекста: общие сведения, теги, атрибуты.

Задание 1

1. Создать папку «Сайт» для размещения в ней страниц *web-сайта* и материалов, которые будут использованы в нашем сайте.
2. Открыть блокнот ввести текст структуры *web-страницы:*

```
<HTML>
    <HEAD>
        <TITLE>Заголовок</TITLE>
    </HEAD>
    <BODY>
        Тело  веб- страницы
    </BODY>
</HTML>
```

1. Выбрать пункт меню Файл →Сохранить как→Имя файла: index.htm
2. Таким же образом создать остальные странички нашего сайта

Примечание: чтоб не прописывать структуру каждой странички, можно пересохранить файл index.htm, открытый в Блокноте, под именами необходимых страниц.

3. У нас получится: index.htm, spisok.htm, frame.htm, zanyatie.htm, dosug.htm, raspisanie.htm, abiturientu.htm, svoya.htm.

При этом:

index.htm – главная страница сайта, содержащая основную информацию

spisok.htm – страница гиперссылок для новигации по сайту

frame.htm – фрейм. Фреймы разбивают веб-страницу на отдельные миникадры, расположенные на одном экране, которые являются независимыми друг от друга. Каждое окно может иметь собственный адрес. При нажатии на любую из ссылок, расположенных в одном фрейме, можно продолжать видеть страницы в других окнах.

zanyatie.htm – страница с информацией о дисциплинах

dosug.htm - страница с информацией о Центре дополнительного образования

raspisanie.htm - страница с информацией расписания

abiturientu.htm - страница с информацией о правилах приема в ВСК

svoya.htm – аторская страница создателя сайта

Задание 2

1. Отрыть папку «Сайт», найти файл **frame.htm,** щелкнуть на нем правой кнопкой мыши и выбрать в контекстном меню Открыть с помощью Блокнот.
2. Отредактировать HTML код страницы:

```
<HTML>
<HEAD>
    <TITLE>  Вологодский строительный колледж  </TITLE>
</HEAD>
    <FRAMESET  cols="20%, 80%">
       <FRAME  SRC=" spisok.htm">
       <FRAME   name=frame2>
    </FRAMESET>
</HTML>
```

Примечание: значение – оформление сайта будет представлено двумя областями – фреймами. В левом небольшом по размеру фрейме находится набор ссылок, при нажатии на них соответствующие документы будут открываться в правом фрейме. Для этого правому фрейму дается имя <FRAME name=frame2>, и оно указывается в ссылках в атрибуте target<ahref=urltarget= frame2>.

Задание 3

1. Отрыть папку «Сайт», найти файл **spisok.htm,** щелкнуть на нем правой кнопкой мыши и выбрать в контекстном меню Открыть с помощью Блокнот.
2. Отредактировать HTML код страницы:

```
<HTML>
<HEAD>
<TITLE>  Вологодский строительный колледж  </TITLE>
</HEAD>
<BODY BGCOLOR=AQUA>
<FONT text=red size=5>
<P><a href= index.htm target= frame2>Информация о ВСК </a>
      <P><a href= zanyatie.htm target= frame2>Учебныедисциплины</a>
```

<P><ahref= dosug.htmtarget= frame2> Центр дополнительного образования
<P>Расписаниезанятий
<P>Информациядляпоступающих
<P>Обавторе
</FONT>
</BODY>
</HTML>

3. Откройте страницу **spisok.htm** в Браузере и проверьте работоспособность ссылок

Практическая работа №16

Размещение на web-странице графических элементов. Организация гиперссылок

6. Работа с текстом. Основные теги.

Следующие теги устанавливают различный уровень заголовков. Они как бы структурируют документ. Фактически они устанавливают различный размер шрифта. Теги парные.

<H1></H1>; <H2></H2>; <H3></H3>; <H4></H4>;<H5></H5>; <Нб></Нб>

<CENTER></CENTER> - центрирует текст, выделенный тегами.

<P> - вводит новый абзац (строку). Обычно закрывающий тег для этого тега не указывают. Тег <P> может иметь атрибут align=left /right /center /justify. Этот атрибут устанавливает выравнивание абзаца, соответственно по левому краю/правому краю/по центру/по ширине. По умолчанию используется выравнивание по ширине. Значения этих атрибутов действуют до следующего тега<P>.

Пример

<PALIGN=CENTER> Весёлые гуси

Текст «Весёлые гуси» будет выровнен по центру.

 - вводит новую строку. Тег не парный. Отличие тега
 от тега <P> заключается в том, что в новой строке сохраняется такое же выравнивание, как и всего абзаца и в теге
 выравнивание не задать. Кроме того между абзацами расстояние больше чем между строками.

<BASEFONT> - Тег непарный. Устанавливает базовый (основной) шрифт для всей страницы.

Атрибуты тега <BASEFONT>

color - цвет шрифта.

size - (1...7) размер шрифта. Этот размер условный, чтобы определить какой размер нужен, необходимо попробовать несколько размеров.

face - шрифт (название шрифта, указывается в кавычках).

Пример:

<BASEFONTCOLOR=REDSIZE=4> - цвет шрифта - красный, размер - 4.

<BASEFONT COLOR-00AA00 FACE="ARIAL"> -цветшрифта - зелёный, шрифт - Arial.

<FONT></FONT> Тег парный. Устанавливает параметры форматирования шрифта для текста, заключенного между тегами. Атрибуты этого тега такие же как и у предыдущего

Пример:

<FONTSIZE=7 COLOR=123456> Компьютерные технологии </FONT>

На странице будет отображаться текст «Компьютерные технологии» размером 7 и цветом 123456.

Следующие теги устанавливают некоторые параметры шрифта, заключенного между тегами. Теги парные.

<B></B> - жирный шрифт.

<I></I> -курсив.

<U></U> - подчеркнутый шрифт

 - соответственно нижний и верхний индекс.

<MARQUEE></ MARQUEE> - бегущая строка.

7. Бегущая строка

Атрибуты тега <MARQUEE>

height - высота в пикселах.

bgcolor –цвет фона.

direction=left /right - направление движения.

behavior=scroll/slide/alternate -способдвижения.

scrollamount -шаг, на который смещается строка (регулирует скорость движения)

scrolldelay - время в миллисекундах, через которое происходит смещение (регулирует скорость движения). Чем больше значение атрибута scrollamount, тем скорость движения выше, чем больше значение атрибута scrolldelay, тем скорость движения ниже.

Пример:

<MARQUEEheight=20 behavior=alternate> Наши поздравления с Вашими успехами </MARQUEE>

По экрану будет двигаться текст «Наши поздравления с Вашими успехами».

Текст внутри бегущей строки можно форматировать, например, так

<MARQUEE><FONTCOLOR=RED> Наши поздравления с Вашими успехами </FONT></MARQUEE>

или так

<FONTCOLOR=RED><MARQUEE> Наши поздравления с Вашими успехами </MARQUEE></FONT>

В любом из последних примеров по экрану будет двигаться красный текст.

8. Списки

Списки могут быть маркированными, нумерованными или иерархическими.

Маркированный список определяется тегом <UL></UL>. Тег парный.

Атрибуты тега <UL>

TYPE=CIRCLE/DISK/SQUARE, определяет вид маркеров (кружок, окружность, квадрат).

Элементы списка определяются тегом <LI>. Тег парный, но закрывающий тег можно не указывать.

Пример:

На странице требуется разместить следующий список:

- Яблоки
- Груши
- Виноград

Этот же список в виде тегов HTML:

<UL>
<LI> Яблоки
<LI> Груши
<LI> Виноград </UL>

Нумерованный список задаётся тегом <OL></OL>. Тег парный.

Атрибуты тега <0L>

TYPE=l/A/a/I/I -определяет вид нумерации (Арабскими цифрами/Большими буквами/ Большими римскими цифрами/Маленькими римскими цифрами)

START=N, где N целое число - атрибут указывает, с какого числа следует начинать нумерацию.

Пример:

На странице требуется разместить следующий список.

1. Круг
2. Квадрат
3. Трапеция
4. Многоугольник

Этот же список в виде тегов HTML.

```
<OL>
    <LI> Круг
    <LI> Квадрат
    <LI> Трапеция
    <LI> Многоугольник
</OL>
```

Специальных тегов для определения иерархического списка в HTML нет. Но всё же возможность создавать такие списки существует.

Пример:

1. Города Вологодской области
    a.Вологда
    b.Череповец
    c.Великий Устюг
2. Города Московской области
    a.Москва
    b.Дедовск
    c.Зеленоград
    d.Руза
    e.Истра

Для создания такого списка нужно использовать следующие теги HTML:

```
<OL>
    <LI>Города Вологодской области
        <OLTYPE=A>
            <LI> Вологда
            <LI> Череповец
            <LI> Великий Устюг
        </OL>
    <LI> Города Московской области
        <OLTYPE=A>
            <LI>Москва
            <LI> Дедовск
            <LI> Зеленоград
            <LI> Руза
            <LI> Истра
        </OL>
</OL>
```

Атрибуты тэга <OL>

COMPACT	*Представляет список в более компактном виде.*
TYPE=A	*Устанавливает маркер в виде прописных букв.*

TYPE=a	*Устанавливает маркер в виде строчных букв.*
TYPE=I	*Устанавливает маркер в виде больших римских цифр.*
TYPE=i	*Устанавливает маркер в виде маленьких римских цифр.*
TYPE=1	*Устанавливает маркер в виде арабских цифр.*
START=n	*Устанавливает начальный маркер в текущем списке.*

Использование разных типов маркеров позволяет лучше выделять один список из другого.

9. Гиперссылки

Гиперссылки служат для связи документов в Web. С помощью гиперссылок происходит навигация по Web-страницам. Щелкнув по гиперссылке можно переходить от одного электронного документа к другому. Совокупность электронных документов с гиперссылками называется гипертекстом. Гиперссылки выделяются цветом или подчеркиванием, в зависимости от настроек браузера.

Для вставки гиперссылки в документ используется тег <a> (Anchor). Href - основной атрибут этого тега. Он показывает на какой файл ссылается эта ссылка.

Пример гиперссылки:

 Абракадабра

В этом примере гиперссылка ссылается на файл «abra.htm», а в тексте web страницы пользователь увидит текст «Абракадабра». В этом примере файл «abra.htm» должен находится в одной папке с документом, в котором расположена эта ссылка.

 Абракадабра

В этом примере, так же как и в предыдущем, на странице пользователь увидит текст «Абракадабра», но «abra.htm» находится в подпапке «sys». Папка sys является подпапкой той папки, в которой находится отображаемая страница.

Последние два примера демонстрируют относительные ссылки. Следующим приведён пример абсолютной ссылки. С помощью этого тега можно также делать ссылки на страницы, расположенные на любом сервере в Интернет.

href-"#pref" - ссылка ссылается на закладку.

href=mailto:box@volnet.ru – ссылка на электронный почтовый ящик (используется слово mailto).

Пример:

<a href="mailto:box@volnet. ru"> Напишите мне

В этом случае пользователь увидит на странице текст «Напишите мне», а при щелчке по этой гиперссылке запустится почтовая программа, откроется окно редактирования сообщения, а строке «Кому» будет введён текст box@vol.net. ru.

Другие атрибуты тега <a>

target="new" - гиперссылка открывается в новом окне.

title - текст всплывающей подсказки гиперссылки.

name - имя закладки.

10. Закладки

Закладки используются в длинных документах, которые не помещаются на экране, для прокрутки. В случае щелчка пользователем по ссылке, указывающей на закладку, другая страница не открывается, а прокручивается открытая страница. Закладка оказывается вверху окна.

Пример использования закладки:

 Принтер

… •
… • •
… • •
 Виды принтеров

В этом примере пользователь видит в окне текст-гиперссыку «Принтер». При щелчке по этой гиперссылке страница будет прокручена до текста «Виды принтеров», причём этот текст будет располагаться вверху окна.

Цвет гиперссылок на странице определяется следующими атрибутами тега BODY: link – цвет гиперссылки, alink - цвет гиперссылки, на которую наведён указатель мыши, vlink - цвет просмотренных гиперссылок. Цвет гиперссылок следует менять в случае крайней необходимости, так как пользователи привыкли, соответственно, к следующим цветам: синий/красный/бордовый (эти цвета используются по умолчанию в браузере InternetExplorer), и могут не понять если ссылки будут отображаться в другом цвете.

Примечание. Здесь и в дальнейшем рекомендуется давать файлам имена, состоящие только из маленьких английских букв. Это связано с тем, что серверы в Internet, как правила работают под управлением операционных систем Unix или Linux, а эти операционные системы, в отличие от Windows, различают большие и маленькие буквы в именах файлов. Например файлы dog.jpg и Dog.jpg будут считаться разными. Поэтому, во избежание путаницы, следует все файлы называть только маленькими английскими буквами. Так же нельзя называть файлы Web-страниц русскими буквами, так как кодировка русских букв в операционных системах Unix и Linuxиная нежели в Windows.

11. Работа с рисунками в HTML

На Web-страница чаще всего используются следующие типы графических файлов. Их различают по расширению.

jpg- обычно используется для хранения фотографий. Это сжатый формат. Файлы в этом формате , обычно, имеют небольшой размер. Сохраняет до 16 млн. цветов.

gif- позволяет сохранять анимированные (движущиеся) изображения. Используется не более 256 цветов. Возможно использование прозрачного фона.

png- усовершенствованный gif - формат, позволяющий сохранять до 16 млн. цветов. Web-страница представляет собой обычный текстовый файл. В этом файле изображения не вставляются. Изображения хранятся в отдельных файлах, а тексте web-страницы хранится только ссылка на графический файл. Если графического файла на месте не окажется, то вместо изображения в браузере пользователь увидит пустой прямоугольник.

<IMG>-тег вставки изображений. Основной атрибут этого тега SRC, он указывает файл изображения.

Простейший пример вставки изображения на Web-страницу.

<IMG SRC="image\101b.jpg">

В данном примере в текст страницы вставляется рисунок "101b.jpg". Обычно рисунки размещаются в отдельных папках. В данном случае файл размещается в подпапке image.

Другие атрибуты тега IMG

align=botton/top/middle/left/right- выравнивание рисунка на странице и взаимное расположение текста и рисунка (botton - текст располагается по нижнему краю рисунка, top - текст располагается по верхнему краю рисунка, middle - текст располагается по середине рисунка, left - текст располагается слева от рисунка, right - текст располагается справа от рисунка). alt="текст" - текст, который будет выводиться вместо рисунка при отключении рисунков в браузере, кроме того это текст всплывающей подсказки.

Значение следующих атрибутов указывается в пикселях

border - толщина рамки.

heightawidht - высота и ширина рисунка. Этот атрибут рекомендуется устанавливать, причём чтобы значения соответствовали реальным размерам рисунка по следующим причинам. Сначала при загрузке страницы загружается текст, а потом уже графические изображения. Если размеры рисунка не указаны, то браузер, до того, как загрузятся рисунки, отображает их в виде произвольных рамок. После загрузки текста начинают загружаться рисунки, причём теперь они отображаются уже в реальном размере вместо маленьких рамок. Получается то, что можно назвать землетрясением: при появлении рисунка на экране текст раздвигается, уступая место рисунку, и текст невозможно читать до окончания загрузки рисунков. Если же размеры рисунков указаны, то в тексте сначала отображаются рамки, соответствующие размерам рисунка, а потом в эти рамки помещаются рисунки, а изменений в тексте не происходит. Теперь почему желательно указывать размеры реальными. Атрибуты **height**и **width**позволяют отобразить рисунок как больше, так и меньше его реального размера, но в случае, если рисунок отобразить маленьким, когда он на самом деле большой, приведёт к увеличению (иногда значительно) времени загрузки страницы, ведь файл рисунка по-прежнему будет большой. Если же рисунок увеличить с помощью этих атрибутов, то рисунок будет низкого качества, хотя незначительное увеличение размера рисунка всё же допустимо, но это нужно смотреть, как говориться, «на месте».

vspace и hspace - расстояние до текста по горизонтали и вертикали.

Пример. Изображие-гиперссылка:

Изображение может быть гиперссылкой, тогда это запишется, например, так

<A HREF="index.htm"><IMG SRC="image\101b.jpg"></A>

В этом случае настраница будет отображаться рисунок **101b.jpg,** находящийся в папке image. Этот рисунок будет гиперссылкой, ссылающейся на файл index.htm,находящийся в этой же папке.

Если нужно вставить рисунок в качестве фона Web-страницы, для этого можно воспользоваться атрибутом BACKGROUNG тега <BODY>.

Пример:

<BODY BACKGROUNG ="texture\al.jpg">.

В этом случае на фон страницы помещается файл "al. jpg". Этот файл находится в подпапке "texture".

Если рисунок маленький, то он просто размножается и копии размещаются рядом.

Ещё один атрибут тега <BODY>bgproperties="fixed" устанавливает, что фон при прокрутке страницы прокручиваться не будет.

Практическая работа №17

Размещение на web-странице таблиц

12. Таблицы

Таблицы в HTML определяются при помощи тега <TABLE>. . . </TABLE>. Тег парный.

Название таблицы можно задать тегом <CAPTION>. . .</CAPTION> (тег парный). Название таблицы отображается непосредственно над таблицей. Здесь можно использовать все теги форматирования шрифта. Можно не использовать тег <CAPTION>, а напечатать текст непосредственно перед таблицей.

Данные в таблице организованы построчно, каждая новая строка таблицы задается тегом <TR>. . . </TR>*{закрывающий тег </TR> можно не использовать).*Строки таблицы разбиваются на ячейки при помощи тегов ячеек-данных <TD>...</TD>*{допускается форма*

*записи без закрывающих тегов).*Для заголовков таблицы можно использовать тег <TH>. . .</TH>. Использование этого тега аналогично тегу <TD>, при этом текст в ячейках выравнивается по центру, а буквы форматируются жирным шрифтом.

Пример использования табличных тегов на простейшей таблице

```
<TABLE>
  <CAPTION  ALIGN=TOP>списокдрузейиподруг</CAPTION>
        <TR>
              < TD> ФИО </ TD>
              < TD>Телефон</ TD>
        </TR>
        <TR>
              <TD> Иванов  Иван  Иваныч </TD>
              <TD>35-35-35</TD>
        </TR>
        <TR>
              <TD> Валина Валентина Валентиновна</TD>
              <TD>46-46-46</TD>
        </TR>
</TABLE>
```

Ниже приведена таблица, которая описывается этими тегами.

Список друзей и подруг

ФИО	**Телефон**
Иванов Иван Иваныч	35-35-35
Валина Валентина Валентиновна	46-46-46

Примечание. В приведённом в примере тегах границы таблицы отображаться не должны. В приведённой таблице границы отображаются только для наглядности.

Атрибуты тегов:

Тег <TABLE>

ALIGN — Общее горизонтальное выравнивание таблицы на странице - LEFT/RIGHT/CENTER (по левому краю/по правому краю/по центру)

VALIGN — Общее вертикальное выравнивание элементов таблицы - BOTTOM/MIDDLE/TOP (по нижнему краю/по середине/по верхнему краю, по умолчанию выравнивание по середине)

CELLPADDING — Свободное пространство между содержимым ячеек и их границами

CELLSPACING — Свободное пространство между границами смежных ячеек

WIDTH — Ширина таблицы в пикселях или процентах ширины окна браузера.HEIGHT — Высота таблицы в пикселях

BORDER — Толщина рамки таблицы в пикселях (по умолчанию рамки нет, BORDER=0) BGCOLOR — Цвет фона таблицы

Тег <TR>

ALIGN — Общее выравнивание элементов строки — LEFT/RIGHT/CENT (по левому краю/по правому краю/ по центру)

Теги <TD> и <TH>

ALIGN — Выравнивание по горизонтали данных в ячейке — LEFT/RIGHT/CENTER (по левому краю/по правому краю/по центру).

VALIGN —Выравнивание заголовка – BOTTOM/MIDDLE/TOP (по верхнему краю/по середине/по нижнему краю).

WIDTH — Ширина ячейки таблицы в пикселях или процентах от ширины таблицы.
HEIGHT — Высота ячейки таблицы в пикселях или процентах от ширины таблицы.
ROWSPAN — Объединение ячеек соседних строк *(ROWSPAN=N)*
COLSPAN — Объединение ячеек соседних столбцов *(COLSPAN=N)*

Пример

Пример таблицы с объединёнными ячейками

1 столбец	**2 столбец**	**3 столбец**
Простая ячейка	Объединенная ячейка	
Простая ячейка	Простая ячейка	Объединенная ячейка
Простая ячейка	Простая ячейка	

Этот же пример в тегах HTML:

```
<HTML>
    <BODY>
        <TABLE BORDER=3 align=center>
            <CAPTION><H1>Примертаблицыобъединённымиячейками</H1></C
        APTION>
                <TR ALIGN=CENTER>
                    <TH> 1столбец</TH>
                    <TH> 2 столбец </TH>
                    <TH> 3 столбец </TH>
                </TR>
                <TR align=center>
                    <TD>Простаяячейка</TD>
                    <TD                                    C0LSPAN=2
                    height=100>Объединённаяячейка</TD>
                </TR>
                <TR ALIGN=CENTER>
                    <TD>Простаяячейка</TD>
                    <TD height=50>Простаяячейка</TD>
                    <TD ROWSPAN=2>Объединённаяячейка</TD>
                </TR>
                <TR ALIGN=CENTER>
                    <TD>Простаяячейка</TD>
                    <TD height=50>Простаяячейка</TD>
                </TR>
        </TABLE>
    </BODY>
</HTML>
```

13. Использование цветов

В HTML не предусмотрено специальных средств раскрашивания таблиц. Однако как NetscapeNavigator, так и MicrosoftInternetExplorer - имеют расширения, позволяющие изменять цвет ячеек и рамок. Вы можете изменить цвет фона ячейки при помощи атрибута BGCOLOR перед размещением в ней текста или изображения, а также использовать атрибут BORDERCOLOR для изменения цвета рамки ячейки.

Тэги <TABLE>, <TD>, <TH> и <TR> допускают использование в них указанных атрибутов. Таким образом, вы можете изменить цвет всей таблицы, отдельной ячейки или строки таблицы.

Примечание. В HTML используются следующие стандартные имена цветов (справа даны их шестнадцатеричные эквиваленты):

Black	(Чёрный)	000000	Silver	(Серебряный)	#C0C0C0
Maroon	(Темно-Бордовый)	800000	Red	(Красный)	#FF0000
Green	(Зеленый)	#008000	Lime	(Известь)	#00FF00
Olive	(Оливковый)	#808000	Yellow	(Желтый)	#FFFE00
Navy	(Темно-Синий)	#000080	Blue	(Синий)	#0000FF
Purple	(Фиолетовый)	#800080	Fuchsia	(Фуксия)	#FF00FF
Teal	(Чирок)	#008080	Aqua	(Аква)	#00FFFF
Gray	(Серый)	#808080	White	(Белый)	#FFFFFF

Практическая работа №18

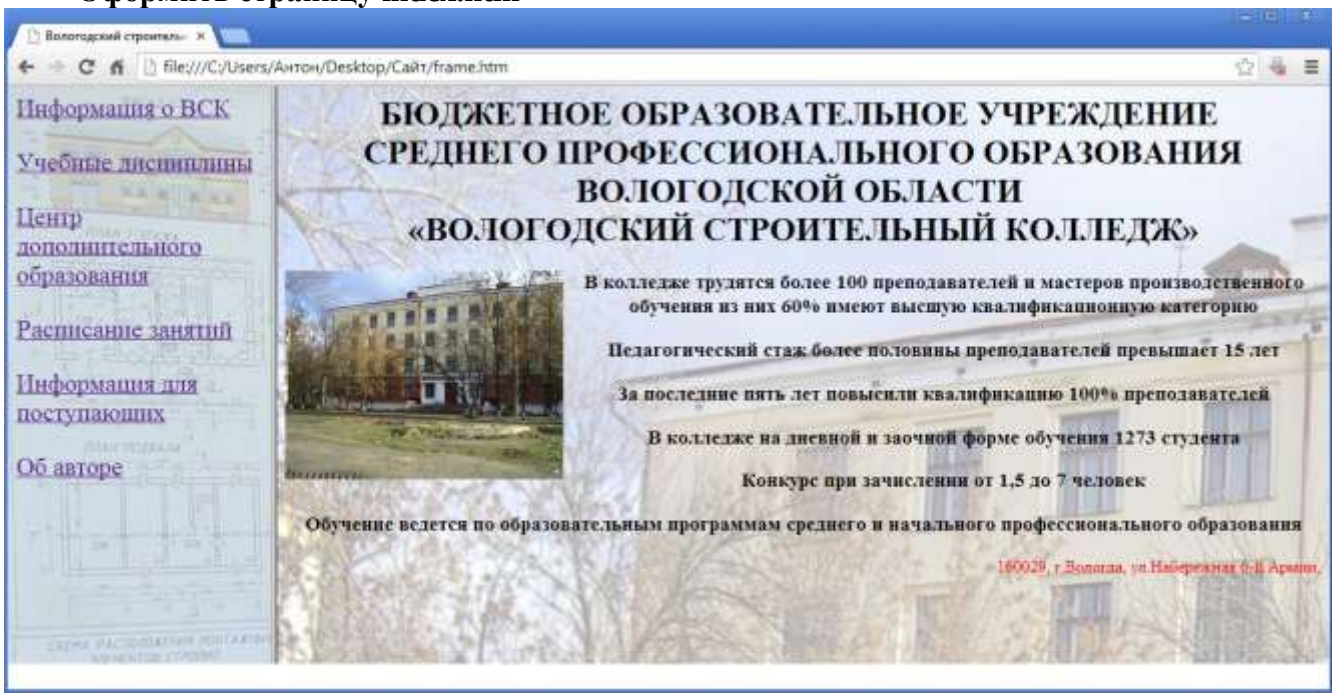
Создание web-страниц на основе фреймов

Задание 1

Установить фон, используя файл fon2.jpg, на странице spisok.htm

Задание 2

Оформить страницу index.htm



Примечание:

1. Фон страницы – файл fon.jpg
2. Записать заголовок для отображения в строке Заголовка Браузера <TITLE> Вологодской строительный колледж </TITLE>
3. Заголовок на странице оформить тегами <H1></H1>
4. Вставить картинку – файл fasad.jpg, обязательно указать высоту и выравнивание по левому краю
5. Бегущая строка внизу страницы «160029, г.Вологда, ул.Набережная 6-й Армии, 199;тел.: 8(8172) 27-02-53,27-32-35; volsk.dir@mail.ru; www.vologda-

vsk.ru»

Задание 3

Оформить страницу zanyatie.htm

1. Установить фон страницы, используя файл fon.jpg
2. Записать заголовок для отображения в строке Заголовка Браузера <TITLE> Изучаемые дисциплины </TITLE>
3. Оформить заголовок на странице «Изучаемые дисциплины» тегами <H1></H1>
4. Перечислить изучаемые дисциплины – оформить в виде маркированного списка

Задание 4

Оформить страницу dosug.htm

1. Установить фон страницы, используя файл fon.jpg
2. Записать заголовок для отображения в строке Заголовка Браузера <TITLE> Центр дополнительного образования </TITLE>
3. Оформить заголовок на странице «Центр дополнительного образования» тегами <H1></H1>
4. Оформить подзаголовок страницы «В Центре дополнительного образования занимаются более 500 студентов» тегами <H2></H2>
5. Информацию оформить в виде таблицы:

dosug1.jpg	Секции по баскетболу, волейболу
dosug2.jpg	Стрельба
dosug3.jpg	Шахматный клуб
dosug4.jpg	Тренажерный зал
dosug5.jpg	Вокал, гитара, танцы

При этом задать выранивание всей таблицы по центру, ширину таблицы – 60% от экрана Браузера, горизонтальное выравнивание содержимого ячеек – по центру. Задать размер текста в ячейах = 5: <FONT SIZE=5>, ширину картинок = 200: width=200.

Задание 5

Оформить страницу raspisanie.htm

1. Установить фон страницы, используя файл fon.jpg
2. Записать заголовок для отображения в строке Заголовка Браузера <TITLE> Расписание </TITLE>
3. Оформить заголовок на странице «Расписание занятий» тегами <H1></H1>
4. Оформить подзаголовок страницы «Расписание занятий с «__»_____20__г. по «__»_____20__г.» тегами <H2></H2>
5. Расписание оформить в виде таблицы, обязательно применить Границы (BORDER, по умолчанию BORDER=0).

Задание 6

Оформить страницу abiturientu.htm по образцу

(Установить фон страницы, используя файл fon.jpg, вставить картинки: abit1, abit2)

Образец:

Правила приема в ВСК

Прием документов

дневное обучение
на базе 9 классов – с 14 июня по 22 июля
на базе 11 классов - с 14 июня по 15 августа
заочное обучение
с 14 мая по 15 сентября

При подаче заявления о приеме на поступающий предъявляет следующие документы:

- документ, удостоверяющий личность (паспорт)
- оригинал (или копия) документа об образовании
- оригинал (или копия)свидетельства о результатах ЕГЭ, ГИА
- фотографии– 4шт. (3 х 4 -матовые)

Другие документы могут быть представлены поступающим, если он претендует на льготы, установленные законодательством РФ, или затребованы от поступающего при наличии ограничений на обучение по соответствующим специальностям и профессиям колледжа, установленным законодательством РФ.

Дополнительно для поступающих на заочное обучение :

- копия трудовой книжки или справка с места работы

- копия свидетельства о браке (если фамилии в паспорте и документе об образовании не совпадают)

- почтовый конверт с маркой

Специальности СПО

Дневное обучение

Наименование специальности	Квалификация	Срок обучения	Уровень предшествующей подготовки	КЦП
«Сервис домашнего и коммунального хозяйства	Специалист по домашнему и коммунальному хозяйству	2г 10 м	среднее (полное) общее	25
«Строительство и эксплуатация зданий и сооружений	Старший техник	4г 10 м	основное общее	25
«Строительство и эксплуатация автомобильных дорог и аэродромов»	Старший техник	4г 10 м	основное общее	25
«Землеустройство»	Специалист-землеустроитель	4г 6 м	основное общее	25
«Технология деревообработки»	Техник	3г 10 м	основное общее	25

Заочное обучение

«Строительство и эксплуатация зданий и сооружений	Техник	3г 10 м	среднее (полное) общее	25
«Строительство и эксплуатация автомобильных дорог и аэродромов	Техник	3г 10 м	среднее (полное) общее	25
«Монтаж и эксплуатация внутренних санитарно-технических устройств, кондиционирования воздуха и вентиляции »	Техник	3г 10 м	среднее (полное) общее	25

Профессии НПО

Наименование профессии	Профессия, квалификация, разряд	Нормативный срок освоения	Уровень предшествующей подготовки	КЦП
«Сварщик (электросварочные и газосварочные работы)»	Электрогазосварщик,3-4 разряд Электросварщик на автоматических и полуавтоматических машинах , 3-4 разряд	2г 5м	основное общее	25
«Мастер отделочных строительных работ»	Штукатур, 3 -4 разряд Маляр строительный , 3-4 разряд	2г 5 м	основное общее	25
«Мастер общестроительных работ»	Каменщик, 3-4 разряд Электросварщик ручной сварки, 3-4 разряд	2г 5 м	основное общее	25
«Мастер столярного и мебельного производства»	Столяр, 3-4 разряд Сборщик изделий из древесины, 3 -4 разряд	2г 5 м	основное общее	25

Выписка из правил приема

Граждане имеют право получить начальное, среднее профессиональное образование на общедоступной и бесплатной основе в колледже, если образование данного уровня получают впервые.

В случае, если численность поступающих превышает количество мест, финансовое обеспечение которых осуществляется за счет бюджета области, при приеме на обучение по образовательным программам среднего профессионального образования по профессиям и специальностям учитываются результаты освоения поступающими образовательных программ основного общего или среднего (полного) общего образования , указанных в представленных поступающими документах об образовании.

В настоящие правила могут быть внесены изменения на основании новых нормативных документов Министерства образования и науки Российской Федерации.

Задание 7

Оформить страницу svoya.htm

1. Установить фон страницы, используя файл fon.jpg
2. Записать заголовок для отображения в строке Заголовка Браузера <TITLE> Об авторе </TITLE>
3. Оформить заголовок на странице «Об авторе» тегами <H1></H1>

Включить в страницу данные о себе:

1. **ФИО**
2. **Фото**
3. **Специальность**
4. **Группа**
5. **Год поступления**
6. **Год выпуска**
7. **…**

Практическая работа №19

Создание HTML-форм

На сайтах практически невозможно обойтись без получения информации от посетителей. Для этого используются формы. Формы позволяют организовать диалог с пользователем, используя такие элементы, как текстовые поля, кнопки, флажки, переключатели, списки, файловые и текстовые поля.

!Форма создается с помощью парного тега <FORM> ... </FORM>, внутри которого помещается все содержимое формы, т. е. элементы управления. Атрибут ACTION задает адрес серверной программы, ENCTYPE — кодировку, а METHOD — метод отправки данных.

Чтобы поместить на Web-страницу форму, проще всего воспользоваться панелью объектов. Для этого нужно переключиться на вкладку Forms, где находятся кнопки, помещающие на Web-страницу форму или один из предусмотренных в HTML элементов управления. Кнопка, помещающая на страницу форму, показана на рис.1 и называется Form. Также можно выбрать пункт Form в меню Insert.

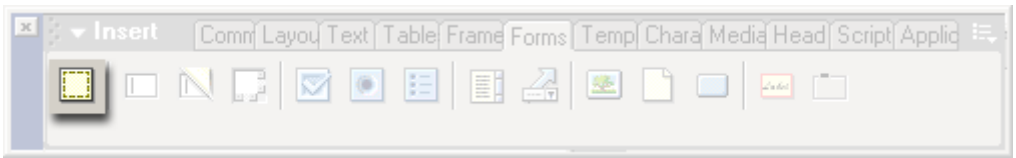


Рис.1. Кнопка Form панели объектов

Пустая форма показывается в виде узкой полоски, ограниченной красной пунктирной рамкой (рис.2). Когда вы поместите что-нибудь в форму, ее размеры соответственно увеличатся.

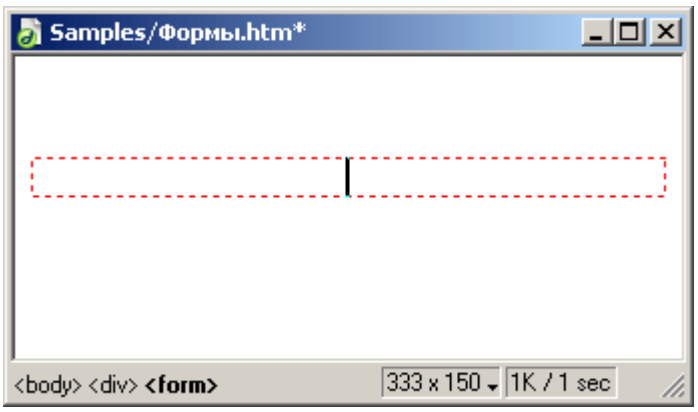


Рис.2 Пустая форма на Web-странице

После создания формы, в режиме ее выделения, окно редактора свойств принимает вид, показанный на рисунке 3

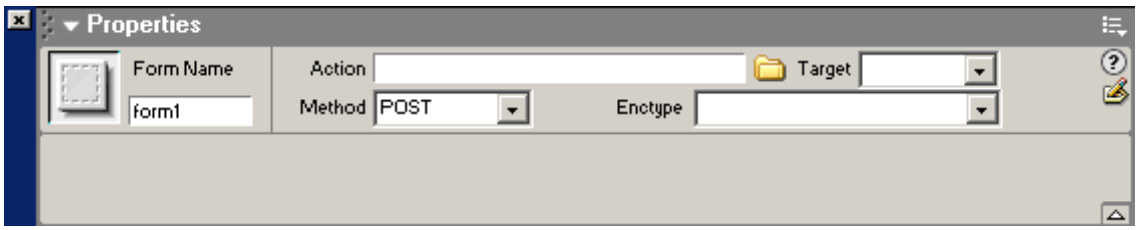


Рис.3 Вид редактора свойств при выделенной форме

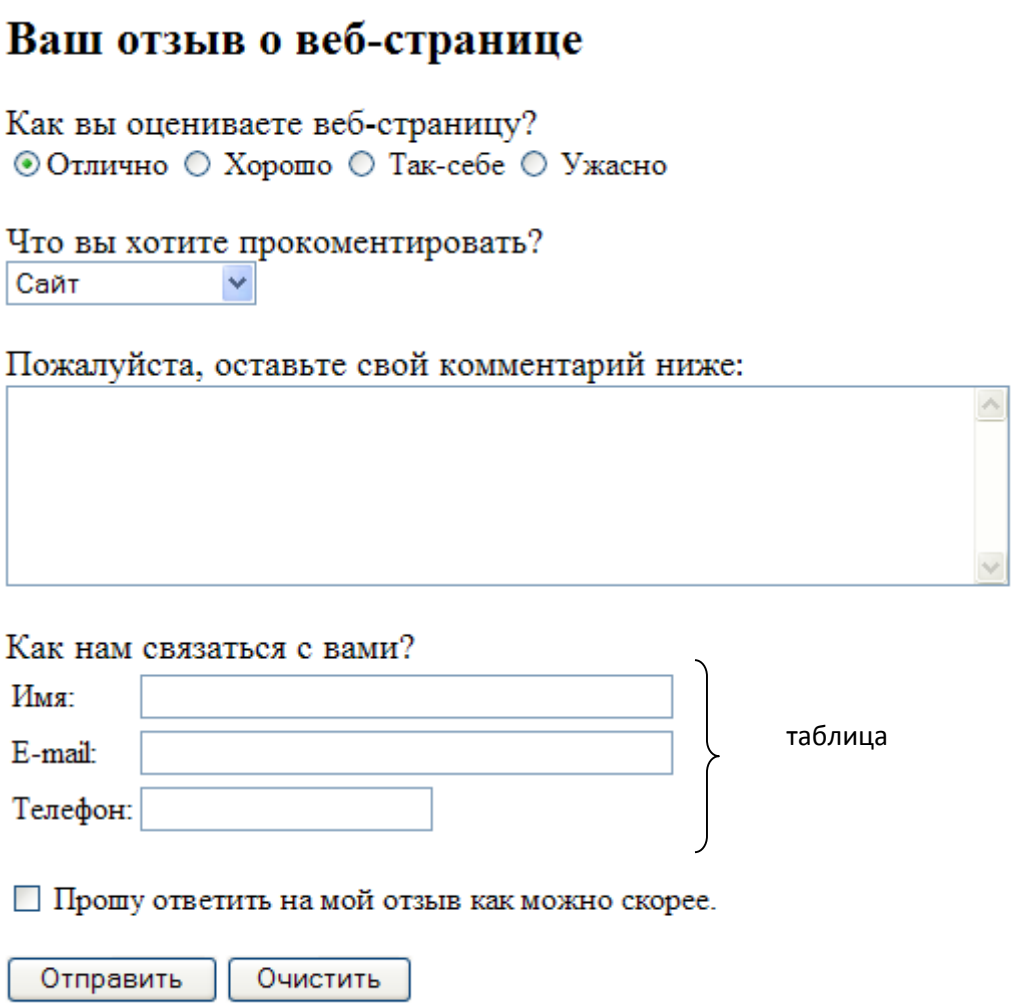
В поле ввода **Form Name** вводится имя формы. По умолчанию Dreamweaver сам подставляет туда автоматически сгенерированное имя вида forml, form2 и т. д. В подавляющем большинстве случаев автоматически сгенерированного имени будет вполне достаточно; вообще, имя формы особой роли не играет, в отличие от имен элементов управления. Имя формы также можно задать, выбрав пункт Name в контекстном меню формы (т. е. контекстном меню, появляющемся при щелчке правой кнопкой мыши по форме)

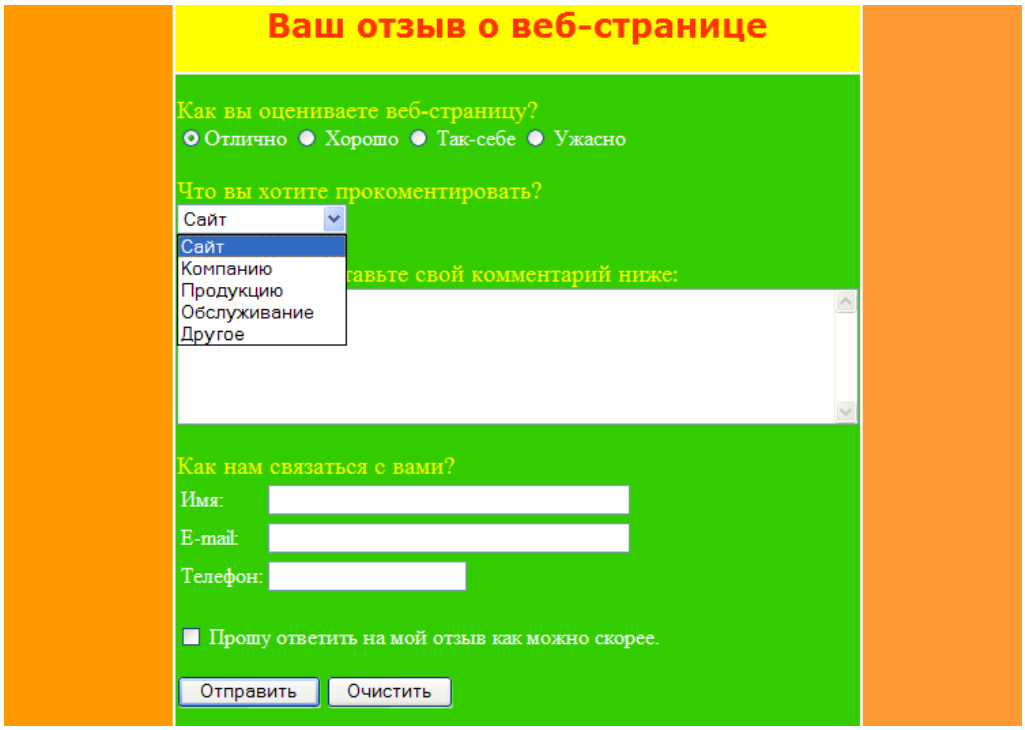
В поле ввода **Action** вводится интернет-адрес серверной программы, которая будет обрабатывать введенные в форму данные. Можно также щелкнуть по значку папки справа от этого поля ввода и выбрать нужный файл в диалоговом окне Select File. Кроме того, можно выбрать пункт Action контекстного меню формы, чтобы вызвать на экран все то же диалоговое окно.

В комбинированном списке **Target** задается цель гиперссылки. Сейчас у нас нет гиперссылки — в данном случае цель задает, куда будет выводиться Web-страница, сгенерированная серверной программой.

Метод пересылки данных задается с помощью раскрывающегося списка **Method.** В этом списке доступны три пункта: GET, POST и Default. Первые два пункта, как вы поняли, задают метод пересылки, а третий — метод, используемый по умолчанию (как правило, GET). Для аналогичных целей служит подменю Method контекстного меню формы, содержащее все те же три пункта.

Задание для самостоятельного выполнения
Средствами Macromedia Dreamweaver создать две HTML-страницы следующей формы и содержания:
Рисунок 4 – форма с таблицей внутри
Рисунок 5 – таблица с формой внутри




Ваш отзыв о веб-странице
Как вы оцениваете веб-страницу?
Отлично
Хорошо
Так-себе
Ужасно
Что вы хотите прокоментировать?
Сайт
Сайт
Компанию
Продукцию
Обслуживание
Другое
тавьте свой комментарий ниже:
Как нам связаться с вами?
Имя:
E-mail:
Телефон:
Прошу ответить на мой отзыв как можно скорее.
Отправить
Очистить

Практическая работа №20
Проектирование дизайна web-узла
Задание. Разработать проект сайта на свободную тему.
Порядок выполнения:

1. РАЗРАБОТКА ЛОГИЧЕСКОЙ СТРУКТУРЫ ВЕБ-САЙТА. Определите тип и количество информации, которая будет помещена на сайт. В результате должно получиться не менее пяти разделов. К полученным разделам обязательно добавьте разделы «Анкета» и «Гостевая книга».
2. ВЫБОР ВИДА ЛОГИЧЕСКОЙ СТРУКТУРЫ САЙТА. Выберите вид логической структуры веб-сайта в соответствии с рекомендациями, данными в разделе 2.1. Предпочтительной является древовидная структура сайта. Подобную структуру имеет сайт, описываемый ниже в качестве примера.
3. ЛОКАЛИЗАЦИЯ ПРОЕКТА. Средствами Windows создайте папку для хранения файлов вашего сайта. Создайте в ней подпапки в соответствии с разработанной физической структурой сайта. При разработке физической структуры рекомендуется создавать отдельную подпапку для каждого раздела.
4. Запустите среду создания веб-сайтов.
5. СОЗДАНИЕ СТРАНИЦ САЙТА. Позаботьтесь о том, чтобы сделать дизайн сайта привлекательным для посетителей, используя предоставленные вам изобразительные средства.
6. ЗАВЕРШЕНИЕ РАБОТЫ. Сохраните получившиеся страницы и перепишите созданные файлы на заранее подготовленный носитель.

Практическая работа №21
Наполнение web-узла контентом
Цель: отработать навыки работы с командами web-программирования.
Задание. Добавить информацию на созданные страницы сайта.
Наполнение сайта включает:

- ✓ подготовка структуры страниц
- ✓ проектирование конверсионных путей
- ✓ разработка иллюстраций и иконок
- ✓ написание текстов для сайта
- ✓ обработка и коррекция изображений
- ✓ корректировка технических параметров
- ✓ программирование слайдеров, веб форм
- ✓ адаптивная верстка страниц
- ✓ заполнение сайта информацией

Практическая работа №22
Разработка навигации web-узла
Цель: отработать навыки работы с командами web-программирования.
Задание. Разработать навигационную структуру сайта.

Навигационная структура сайта требует тщательного обдумывания с самых ранних стадий разработки, ведь при попытках внести изменения в уже готовую структуру сайта практически неизбежна путаница, появление «битых» ссылок. Вот несколько основных правил, касающихся разработки навигационной структуры сайта.

Во-первых, с любой страницы сайта должна быть доступной ссылка на главную (домашнюю) страницу. Зачастую подобная ссылка называется «Главная» или «Home», а

еще практически стандартом в веб-дизайна стало создание активной ссылки на главную страницу на логотипе сайта (ссылка-картинка).

Во-вторых, все ссылки на сайте должны иметь интуитивно понятный посетителям вид. Так, ссылки в контенте страниц сайта традиционно выделяются подчеркиванием и отличным от остального текста цветом, ссылки меню имеют аналогичный вид или же представляют собой кнопки. Если ссылки зрительно нельзя выделить на фоне остальных элементов страниц сайта или же если ссылки имеют непривычный для большинства посетителей вид, то это делает навигацию по сайту крайне неудобной.

В-третьих, структура навигации по сайту должна быть такой, чтобы посетителям в любой момент времени было ясно, на какой странице они находятся, как с этой страницы можно попасть на другие страницы, как вернуться к предыдущей странице. При этом если для выполнения перехода от текущей страницы сайта к какой-либо еще требуется 4, а то и 5-6 кликов мышью, то это говорит о низком уровне юзабилити разработанной навигационной структуры.

Практическая работа №23
Создание макетов страниц web-узла
Цель: отработать навыки работы с командами web-программирования при создании макетов страниц web-узла.

Шаг 1. Создание полного макета веб - сайта начинается с создания его начального расположения, и, только после этого, можно приступать к проектированию индивидуальных элементов страницы. Результат - современная и новая веб - страница - раскладка, готовая к кодированию.

Шаг2. Черпая вдохновение из различных современных проектов веб - сайтов, мы и создадим это новую страницу - раскладку для веб - сайта. Ключевые особенности нашей веб – страницы будут заключаться в горизонтальных полосах, так называемых «группах»: они нам нужны для того, чтобы разместить содержание нашей веб - странички в определенных областях: красочная область заголовка - шапка страницы, будет представлять один участок; дружественное приветственное обращение с примерами работ – другую часть; и, основная часть и ресурс, заполнят нижний колонтитул (нижнюю сноску).

Шаг3. Работа над любым проектом должна начинаться с эскиза на бумаге. При помощи карандаша можно приблизительно или детально создать план будущего расположения частей веб - странички на бумаге.

Шаг4. Планирование каркаса будущей веб – страницы помогает развить иерархию и дает понять, как лучше расположить ключевые моменты проекта.

Шаг5. Создайте новый документ. Будет лучше, если Вы сделаете новый документ размером, соответствующим размеру широкоэкранного монитора, чтобы лучше представить, как будет выглядеть Ваша веб – страничка на широком экране. Например, создайте новый документ шириной – **1680 пикс.,**высотой – **1050 пикс**.

Шаг6. Воспользуйтесь инструментом **Guides«Направляющие»** и разместите эти направляющие линии в центре Вашего нового документа. Ширина направляющих составляет приблизительно **960 пикс**. С помощью этих направляющих, мы создадим основную сетку, для того, чтобы впоследствии нам было удобнее зрительно расположить основные пункты нашей веб – страницы.

Шаг7. Начните работу с создания полосы заголовка: создайте новый слой, выберите

инструмент **Rectangular Marquee Tool «Прямоугольное выделение»**, на новом слое

нарисуйте выделение, похожее на то, что приведено на рисунке ниже и залейте созданное выделение белым цветом.

Далее, войдите в палитру **Layer Style«Стили слоя»**, и добавьте стиль слоя **Gradient Overlay«Наложение градиента»**, цвет градиента выберите от основного к фоновому (от черного к белому) и отрегулируйте**Opacity«Непрозрачность»** на Ваше усмотрение, чтобы эффект градиента получился не очень темным. После всех манипуляций нажмите **«Да»,** чтобы применить эффект**Gradient Overlay«Наложение градиента»**. Градиент рисуйте линейного типа.

Шаг8. Теперь снова создайте новый слой и инструментом **Rectangular Marquee Tool «Прямоугольноевыделение»** нарисуйте область выделения, чуть больше предыдущей: здесь будет располагаться изображаемое содержимое, и в палитре **Layer Style«Стили слоя»** добавьте эффект**Gradient Overlay«Наложение градиента»**. Цвет градиента выберите двух ярких цветов (на Ваше усмотрение); тип градиента: **Linear «Линейный»**. Для того, чтобы добавить немного глубины этому новому слою, в эффектах **Layer Style«Стилислоя»** войдите в графу **Inner Shadow«Внутренняя тень»** и на Ваше усмотрение установите параметры тени.

Шаг9. По желанию: легкая текстура может слегка освежить наш проект веб – страницы. После того, как к нарисованной прямоугольной области Вы применили эффект **Gradient Overlay«Наложение градиента»** из стилей слоя, Вы можете к этому прямоугольнику применить фильтр: **Filter> Noise> Add Noise (Фильтр – Шум – Добавить шум)** и режим наложения поставьте **Multiply«Умножение»**. (если после применения данного фильтра «шума» не видно, попробуйте отключить видимость эффекта**Gradient Overlay«Наложение градиента»** в палитре **Layer«Слои»**).

Шаг10. Добавьте логотип (фирменный знак) компании в той позиции на экране, согласно **Grid«Сетке» - «Направляющим»**, затем добавьте к данному фирменному знаку стиль **Gradient Overlay«Наложение градиента»** и **Inner Shadow«Внутреннюю тень»** из команды **Layer Style«Стили слоя»**. Внутреннюю тень сделайте очень мягкой.

Шаг 11. Используйте инструмент **Type «Текст»** для того, чтобы создать надпись для главной навигации. Установите курсор инструмента**Type«Текст»** в середине серой полосы с градиентом и напишите необходимый Вам текст более темным цветом, чем цвет фона-градиента полоски, на которой пишите.

Шаг12. Особенность заголовка заключается в том, что он занимает большее место на веб - странице. Он является основным центром для пользователя. Используйте вторую двуцветную градиентную полоску для того, чтобы разместить на ней вводный заголовок, написанный другим шрифтом.

Шаг13. Продолжайте к заголовку добавлять детальные пояснения, но, на этот раз, используя шрифты **Arial** или **Helvetica**.

Шаг14. Разместите необходимую Вам иллюстрацию (в нашем случае это будет иллюстрация портативного компьютера). Также, на экране нашего портативного компьютера мы можем разместить примеры работ. Некоторые примеры Вы можете найти на этой ссылке: range of examples can be found here

Шаг15. Теперь, для того, чтобы акцентировать внимание на нашей иллюстрации (портативном компьютере), и зрительно выделить ее, мы создадим новый слой, установим его позади слоя с иллюстрацией и, гармонирующим с синим фоном цветом, добавим прямо в центре радиальный градиент. Яркость его и непрозрачность отрегулируйте на Ваше усмотрение.

Шаг16. Создайте новый слой и на нем, под двуцветной центральной частью с иллюстрацией снова нарисуйте небольшую прямоугольную область и заполните ее серо-белым градиентом.

Шаг17. Разделите среднюю часть страницы на две колонки инструментом **Guides «Направляющие»**относительно линий сетки. В левой части будет размещен основной текст, а в правой части - дополнительная информация, касающаяся

Вашей веб – страницы. Используйте инструмент **Type «Текст»**, чтобы создать надпись в этих двух колонках. Размер и тип шрифта (кегль и гарнитуру) выберите таким образом, чтобы текст был легко читаем.

Шаг18. Ниже этой самой главной части Вашей страницы можно разместить последние новости или блоги. Разделите столбец на два дополнительных столбца и сделайте ссылку на эти блоги. По желанию, можно сделать так, что, проходя по этой ссылке, ее текст будет менять свой цвет.

Шаг19. Создайте новый слой. На нем инструментом **Rounded Rectangle Tool «Прямоугольник соскругленными углами»** нарисуйте именно такой прямоугольник – со скругленными углами. Рисуем в режиме **«Выполнитьзаливку пикселов»** на панели свойств, удерживая при рисовании клавишу**<Shiht>** для того, чтобы прямоугольник получился пропорциональным. Цвет прямоугольнику задаем серый.

Шаг20. После того, как наш прямоугольник нарисован, нам нужно добавить к нему линейный серо-белый градиент, тонкую серую обводку и мягкую внутреннюю тень. Все это мы делаем при помощи команды **Layer Style«Стили слоя»**, которая расположена в нижней части палитры**Layer«Слои».**

Шаг21. Используйте нарисованный и залитый серо-белым градиентом прямоугольник – виртуальную панель, для того, чтобы разместить в ней небольшой скриншот и необходимый текст, относящиеся к основной тематике создаваемой веб – странице.

Шаг22. В нижней части серого прямоугольника на новом слое нарисуйте другой прямоугольник со скругленными углами: он будет выполнять роль «кнопки» на Вашей веб – странице. К этому прямоугольнику добавьте такие **Layer Style«Стили слоя»**, как: наложение градиента и обводку серого цвета. Эти **Layer Style«Стили слоя»** мы добавляем для того, чтобы наша «кнопка» выглядела эффектно и стилизованно.

Шаг23. Напишите на «кнопке» необходимый Вам текст (метку), который будет подсказывать пользователям, каким образом можно дальше продолжать просматривать через веб – страницу очередные проекты.

Шаг24. Обозначьте конец содержимого на экране областью нижнего колонтитула. Это сделайте при помощи новой прямоугольной области, которую создайте на новом слое и залейте светло-серым цветом.

Шаг25. Инструментом **Elliptical Marquee Tool «Овальное выделение»** на новом слое нарисуйте длинный и узкий эллипс и заполните это выделение радиальной градиентной заливкой цветом от черного к прозрачному. При активном слое с нарисованным эллипсом, нажмите комбинацию клавиш**<Ctrl+T>**, либо**:Edit>Transform> Scale(Редактирование – Трансформирование – Масштабирование**) для того, чтобы созданный эллипс масштабировать в длинный и тонкий эллипс, похожий на тень.

Шаг26. Разместите эллипс - тень на экране по центру, затем удалите лишнюю область выше нижнего колонтитула. В результате, у Вас должна получиться тонкая тень, которая добавит небольшой объем деталям.

Шаг27. Область нижнего колонтитула содержит достаточно свободного места для размещения вторичных элементов страницы. Одним из таких элементов может стать **«Область клиентского логина»**. Для того,чтобы создать данную область,

активизируйте инструмент **Type «Текст»,** создайте два (или несколько) текстовых рамок, внутри которых и разместите Ваш текст: логин; пароль. Созданным текстовым рамкам можно придать небольшую мягкую тень, используя функцию: **Layer Style«Стили слоя».**

Шаг28. Используйте центральную область нижнего колонтитула для того, чтобы разместить информацию о компании. Создайте текст, используя легко читаемый шрифт.

Шаг29. Теперь, добавьте контактную информацию крупным читаемым шрифтом.

Шаг30. В итоге, после всей проделанной работы, у нас получилась интересная веб – страничка. Все элементы четко расположены на странице, все удобочитаемо и красиво!

Практическая работа №24
Разработка CSS-правил для управления оформлением web-узла
Цель: получить навыки разработки CSS-правил для управления оформлением web-узла

- При создании правила для нескольких селекторов помещайте каждый селектор на отдельной строке.
- Перед открывающей скобкой ставьте один пробел.
- Внутри блока объявлений помещайте каждое объявление на отдельной строке.
- Добавляйте один уровень отступов перед каждым объявлением.
- Ставьте пробел после двоеточия внутри объявления.
- Всегда ставьте точку с запятой после последнего объявления в блоке.
- Ставьте закрывающую скобку на одной вертикальной линии с первым символом в селекторе.
- Ставьте пробел после каждой запятой в объявлениях со множественным значением.
- Разделяйте правила пустой строкой.
- Сортируйте свойства по принципу: свойства, сильно влияющие на элемент, в начале, а самые незначительно влияющие - в конце:
    - **Display**
    - Позиционирование (**position**/float)
    - Боксовая модель (**width**/**height**/**margin**/**padding**/**border**/**box-sizing**)
    - Цвета и типографика
    - Остальное

Плохо:

```
.crm-lead-form { margin: 34px; color: #000; }
.crm-lead-title, .crm-invoice-title, .crm-company-title
{
```

```
    position: relative;
    background: #000;
    height: 15px;
    padding: 10px;
    border: 1px solid red;
    margin: 12px 0 17px;
    display: block;
    color: #fff;
    width: 15px;
}
```

Хорошо:

```
.crm-lead-form {
    margin: 34px;
    color: #000;
}

.crm-lead-title,
.crm-invoice-title,
.crm-company-title {
    display: block;
    position: relative;
    width: 15px;
    height: 15px;
    padding: 10px;
    border: 1px solid red;
    margin: 12px 0 17px;
    color: #fff;
    background: #000;
}
```

- Для значений с пробелами (**font-family**, **url()**) и для свойства **content** используйте двойные кавычки.

```
.disk-user-profile {
    font-family: "Helvetica Neue Light", Helvetica, Arial, sans-serif;
}

.disk-user-profile:after {
    display: block;
    content: "";


}
```

- Исключения

К большим группам правил, состоящих из одного свойства, может применяться запись в одну строку. В таком случае следует ставить пробел после открывающей и перед закрывающей скобками.

```
.crm-column-title { width: 10%; }
.crm-column-author { width: 20%; }
.crm-column-actions { width: 30%; }

.menu-icon-create { background-position: 0 0; }
.menu-icon-delete { background-position: -15px -35px; }
.menu-icon-approve { background-position: -34px -35px; }
```

Длинные значения свойств, разделяемые запятыми - как, например, набор градиентов или теней - могут быть помещены на отдельной строке каждое, чтобы повысить читабельность кода и сообщений в системе управления версиями. Формат записи может слегка различаться, один из вариантов приведён ниже.

```
.disk-info-popup {
   box-shadow:
      1px 1px 1px #000,
      2px 2px 1px 1px #ccc inset;
   background-image:
      linear-gradient(#fff, #ccc),
      linear-gradient(#f3c, #4ec);
}
```

Практическая работа №25
Разработка XML-документов

Цель: рассмотреть методы разработки XML-документов

Класс **XmlTextWriter**, который является производным от класса **XmlWriter**, записывает XML-данные в файл, консоль, поток, а также может использовать другие типы вывода. При записи XML-данных эти методы выполняют дополнительную работу, чтобы сформировать XML-документы правильного формата. В следующей таблице представлен список методов, которые выполняют работу для обеспечения формирования данных правильного формата.

Метод	**Описание выполняемой работы**
WriteAttributeString	Класс **XmlTextWriter** экранирует текстовое содержимое атрибута в зависимости от того, что он обнаруживает.
WriteString	Класс **XmlTextWriter** экранирует специальные символы, заменяя их при необходимости на **& < >**и числовые символьные сущности.

WriteBase64	Класс **XmlTextWriter** кодирует байты base64, которые затем можно будет прочитать с помощью метода ReadBinary класса XmlReader.

Следующие дополнительные задачи выполняются классом **XmlTextWriter**, чтобы обеспечить формирование XML-документов правильного формата.

- Гарантирует, что XML-элементы записаны в правильном порядке. Например, он не позволит записать атрибут за пределами элемента, записать блок CDATA внутри элемента или записать несколько корневых элементов. Кроме того, он гарантирует, что декларация <?xml следует первой, а узел <!DOCTYPE стоит перед корневым элементом.
- Гарантирует, что значение и формат атрибута xml:space правильные, а также соответствуют рекомендации консорциума W3C по языку XML 1.0 (второй выпуск). В следующем примере показано использование допустимого значения для атрибутаxml:space в методе **WriteAttributeString**:
- w.WriteAttributeString("xml:space", "", "preserve");

  Допустимыми значениями для атрибута xml:space являются **default** и **preserve**. Если аргумент не является одним из этих значений, возникает исключение **ArgumentException**.
- Проверяет, используется ли строка в качестве параметра (например, Null==String.Empty и String.Empty)), а также соответствует ли она правилам W3C.

В следующей таблице приведены определяемые классом **XmlTextWriter** дополнительные методы и свойства, которые не унаследованы от класса **XmlWriter** (и не определены в нем), а также не унаследованы от класса **Object**.

Метод или свойство	**Описание**
XmlTextWriter	Создает экземпляр класса **XmlTextWriter**, который принимает имя файла, поток или объект **TextWriter**. Существует перегруженный метод, принимающий дополнительный параметр, который определяет тип кодирования.
Свойство Namespaces	Указывает, поддерживаются ли пространства имен. Если это свойство имеет значение **false**, декларации **xmlns**не пишутся, и можно указывать имена элементов, содержащие любое количество двоеточий.
Свойство Formatting	Определяет, используются ли отступы для форматирования выходных данных.
Свойство IndentChar	Определяет, какой символ должен использоваться для создания отступов при формировании отступов с помощью свойства **Formatting**.
Свойство Indentation	Определяет, сколько символов **IndentChars** записывать для каждого уровня в иерархии при формировании отступов с помощью свойства **Formatting**.

Свойство QuoteChar	Определяет, какой символ используется для заключения значений атрибутов в кавычки. Это должна быть либо одинарная кавычка **'**, либо двойная кавычка **"**.
BaseStream	Возвращает строку, в которую записывает класс **XmlTextWriter**. Возвращает значение NULL, если класс**XmlTextWriter** был создан с объектом **TextWriter**, который не является производным от объекта**StreamWriter**.

Следующий пример создает выходные XML-данные с помощью класса **XmlTextWriter**.
C#
VB

```
static void WriteQuote(XmlWriter writer, string symbol,
            double price, double change, long volume)
{
  writer.WriteStartElement("Stock");
  writer.WriteAttributeString("Symbol", symbol);
  writer.WriteElementString("Price", XmlConvert.ToString(price));
  writer.WriteElementString("Change", XmlConvert.ToString(change));
  writer.WriteElementString("Volume", XmlConvert.ToString(volume));
  writer.WriteEndElement();
}

public static void Main(){
   XmlTextWriter writer = new XmlTextWriter(Console.Out);
   writer.Formatting = Formatting.Indented;
   WriteQuote(writer, "MSFT", 74.125, 5.89, 69020000);
   writer.Close();
}
```

Выходные данные

```
<Stock Symbol="MSFT">
    <Price>74.125</Price>
    <Change>5.89</Change>
    <Volume>69020000</Volume>
</Stock>
```

Выходными данными для метода **WriteQuote** является символ акции, который метод получает в формате **string**. Цена и изменение декларируются как тип **double**, а объем — как тип **long**. Класс **XmlConvert** используется, чтобы преобразовать эти переменные в строки. Он имеет методы, которые преобразуют все строгие типы данных в строки. Кроме того, класс **XmlConvert** имеет методы, выполняющие обратное преобразование путем преобразования строк в типы данных .NET Framework. Дополнительные сведения см. в разделе Кодирование символов в именах XML и преобразование типов XML-данных.
Образец кода, демонстрирующий запись XML-данных в файл, см. в разделе XmlTextWriter.WriteProcessingInstruction. Образец кода, демонстрирующий запись XML-данных в консоль, см. в разделе XmlTextWriter.WriteString.
Следующий код показывает, как записать элемент, который формирует <price>19.95</price>:
C#

VB

```
//Write the price.
writer.WriteElementString("price", "19.95");
```

Следующий код показывает, как записать атрибут, который формирует <element name="purchaseOrder"/>:

C#

VB

```
writer.WriteStartElement("element");
writer.WriteAttributeString("name", "purchaseOrder");
writer.WriteEndElement();
```

Запись с помощью метода WriteAttributeString атрибутов и деклараций пространств имен

У метода **WriteAttributeString** две разные задачи. Одна заключается в записи атрибутов и ассоциировании их с определенными пользователями префиксами пространств имен. Другая задача состоит в формировании деклараций пространств имен. Если при записи атрибутов параметр localname имеет значение *xmlns*, считается, что этот метод создает декларацию пространства имен.

В следующем примере кода метод **WriteAttributeString** используется, чтобы записать атрибуты внутри элемента.

C#

VB

```
//Write the genre attribute.
writer.WriteAttributeString("genre", "novel");
//Write the ISBN attribute.
writer.WriteAttributeString("ISBN", "1-8630-014");
```

Метод **WriteAttributeString** также экранирует текстовое содержимое атрибута в зависимости от того, что он обнаруживает. При использовании двойных кавычек класс **XmlTextWriter** экранирует их в текстовом содержимом значения атрибута с помощью". При использовании одинарных кавычек он экранирует текстовое содержимое значения атрибута с помощью '.

Чтобы формировать декларации пространств имен, существует перегруженный метод **WriteAttributeString**, позволяющий приложению определять декларацию пространства имен. Следующий пример кода создает два пространства имен по умолчанию. Первая декларация привязывает все элементы без префикса к первой декларации пространства имен, а элементы, декларированные с префиксом «po», привязываются ко второй декларации пространства имен.

C#

VB

```
// Write the default namespace, identified as xmlns with no prefix
writer.WriteAttributeString("xmlns", null, "http://www.w3.org/2000/10/XMLSchema");
// Write a namespace for the purchase order with a prefix of "po"
writer.WriteAttributeString("xmlns", "po", null, "http://contoso.com/po");
```

Метод Close

Метод **Close** проверяет допустимость XML-документа при закрытии потока. Это предотвращает создание недопустимых XML-документов и гарантирует, что XML-документ будет правильного формата. Помимо закрытия потока метод **Close** также вызывает все необходимые методы **WriteEnd<xxx>**, чтобы закрыть документ.

Пары методов

В классе **XmlWriter** также есть пары методов: **WriteStartDocument** и **WriteEndDocument**, **WriteStartElement** и**WriteEndElement**, а также **WriteStartAttribute** и **WriteEndAttribute**. При помощи этих методов можно, например, создать вложенные элементы или атрибуты. Именно с помощью этих пар методов и строится XML-документ. Они позволяют создавать сложные элементы и атрибуты.

Методы WriteStartDocument и WriteEndDocument

Метод **WriteStartDocument** начинает новый документ и записывает XML-декларацию с атрибутом версии, имеющим значение «1.0», а метод **WriteEndDocument** закрывает документ. Перед вызовом следующего метода **WriteStartDocument** для начала записи следующего документа можно изменить форматирование, отступы и другие свойства. Метод **WriteStartDocument**программно распознает записываемый XML-документ и применяет правила корневого уровня. Если этот метод не используется, создается XML-фрагмент и проверяется, что он правильного формата. Правила корневого уровня не применяются. Следующий пример кода показывает начало и конец в документе.

C#

VB

```
// Write the XML declaration.
writer.WriteStartDocument();
. . .
// Close the document.
writer.WriteEndDocument();
```

Методы WriteStartElement и WriteEndElement

Пара методов **WriteStartElement** и **WriteEndElement** разграничивает один или несколько элементов. Во всех переопределенных методах **WriteStartElement** локальное имя для открывающего тега является обязательным параметром. В следующем коде используется пара методов **WriteStartElement** и **WriteEndElement**.

C#

VB

```
// Write the title.
writer.WriteStartElement("title");
writer.WriteString("The Handmaid's Tale");
writer.WriteEndElement();
```

Выходные данные

```
<title>The Handmaid's Tale</title>
```

Метод **WriteStartElement** предоставляет переопределенную подпись метода, которая позволяет коду указывать префиксы пространств имен для своих элементов.

Методы WriteStartAttribute и WriteEndAttribute

Методы **WriteStartAttribute** и **WriteEndAttribute** похожи на другие методы начала и конца, только эти методы начинают и завершают атрибуты. Метод **WriteStartAttribute** записывает начало атрибута, метод **WriteString** используется для записи значения атрибута, а метод **WriteEndAttribute** завершает тег атрибута. В следующем примере кода показана пара методов**WriteStartAttribute** и **WriteEndAttribute**.

C#

VB

```
writer.WriteStartAttribute(prefix, "ISBN", "urn:samples");
writer.WriteString("1-861003-78");
writer.WriteEndAttribute();
```

Выходные данные
<book bk:ISBN="1-861003-78">

Метод **WriteStartAttribute** имеет перегруженный метод, который позволяет приложению указывать префикс пространства имен с тем, чтобы оно могло ассоциировать этот префикс пространства имен с теми атрибутами, которые оно записывает.

Практическая работа №26
Создание простейших сценариев

Создание простых сценариев

Как вы уже знаете JavaScript - это язык подготовки сценариев для вэб-документов. Команды JavaScript вставляются напрямую в документ HTML, и сценарий выполняется непосредственно при загрузке его в браузере.

Инструменты создания сценариев

Вам нет необходимости приобретать специальное программное обеспечение для создания сценариев JavaScript. Все что для этого необходимо - это текстовый редактор. Подойдет любой, если вы еще не определились, пользуйтесь БЛОКНОТОМ, который входит в состав Windows.

Отсчет времени

Одно из привычных использований JavaScript - это отображение даты и времени. Поскольку сценарий JavaScript выполняется в браузере, необходимо время отображать в соответствии с временным поясом места жительства пользователя. Для начала создадим сценарий вычисления всеобщего скоординированного времени. (*Всеобщее скоординированное время соответствует времени нулевого меридиана или времени по Гринвичу. Это время абсолютного уровня отсчета дат.*)

Начало сценария

Наш сценарий, как и большинство других программ JavaScript, начнем с дескриптора <SCRIPT>.

Откройте текстовый редактор (БЛОКНОТ) и введите пару дескрипторов <SCRIPT>:

```
<SCRIPT LANGUAGE="JavaScript">
</SCRIPT>
```

Добавление операторов JavaScript

Чтобы успешно создать сценарий, необходимо сначала правильно определить местное и абсолютное время, а затем отобразить их на вэб-странице. Преобразование времени из одного формата в другой осуществляется исключительно встроенными средствами JavaScript.

Сохранение данных в переменных

В самом начале сценария необходимо определить переменную, в которой будет сохраняться текущая дата и дата начала нового тысячелетия. Детально о переменных речь пойдет дальше, пока воспринимайте их как контейнеры, содержащие определенную информацию.

После дескриптора <SCRIPT> введите приведенные ниже строки. Обратите внимание на использование строчных и прописных букв - JavaScript не только их различает, но и использует по-разному:

```
now = new Date();
```

Этот оператор создает переменную now, которая принимает значение текущей даты. Этот оператор и другие, используемые в сценарии, построены на основе объекта Data, необходимого для задания дат.

В конце каждой строки программы JavaScript ставится разделитель - точка с запятой. Именно он указывает браузеру на конец строки программы.

Вычисление значения

JavaScript определяет дату, как количество миллисекунд, отсчитанных от 1 января 1970 года. Но это не значит, что вам нужно в ручную высчитывать и переводить время, для этого есть встроенные функции.

Чтобы завершить сценарий, перед закрывающим дескриптором </SCRIPT> введите следующие две строки:

```
localtime=now.toString();
utctime=now.toGMTString();
```

Эти операторы создают две новые переменные: localtime и utctime. Первая содержит текущее время и дату, а вторая - их абсолютные значения.

В конечном счете обе переменные содержат текстовые значения. Например, January 25, 2004 12:00 PM. Выражаясь языком программистов, сохраненный в переменной текст называется строкой. Детально о строках поговорим позже.

Вывод результата на экран

Теперь, когда есть две переменные, принимающие требуемое значение в секундах, самое время побеспокоиться о выводе результата на экран. В JavaScript существует несколько способов отображения данных на экране. Один из самых простых заключается в применении операторадocument.write.

Этот оператор позволяет отображать текст, числа и другие типы данных. Поскольку создаваемый сценарий планируется вставить в вэб-страницу, нужно озаглавить рассчитанное значение:

```
document.write("<b>Текущее время: </b>" + localtime + "<BR>");
document.write("<b>Абсолютное время: </b>" + utctime + "</p>");
```

Эти операторы указывают браузеру вывести введенный в кавычках текст, а после него значение соответствующей переменной. Обратите внимание на символ "+", введенный между отображаемым текстом и переменными. В данном случае он указывает на отображение текста и значения переменной в одной строке. Если использовать этот символ между двумя переменными, то их значения будут просто суммироваться и на экране отобразится их сумма.

Вставка сценария на вэб-страницу

Теперь есть готовый сценарий, готовый для вставки на вэб-страницу. Сначала проверьте код сценария, он должен быть таким:

```
<SCRIPT LANGUAGE="JavaScript">
now = new Date();
localtime=now.toString();
utctime=now.toGMTString();
document.write("<b>Текущее время: </b>" + localtime + "<BR>");
document.write("<b>Абсолютное время: </b>" + utctime + "</p>");
</SCRIPT>
```

Для того чтобы выполнить сценарий, необходимо вставить его в код HTML-документа. Самый простой документ HTML содержит элементы HTML, HEAD и BODY. Добавьте необходимые теги и заголовок вэб-страницы к имеющемуся сценарию и получится вполне загружаемый в браузер документ HTML:

```
<HTML>
<HEAD><TITLE>Отображение даты</TITLE>
<BODY>
<H1>Текущее время</H1>
<p>
<SCRIPT LANGUAGE="JavaScript">
now = new Date();
localtime=now.toString();
utctime=now.toGMTString();
document.write("<b>Текущее время: </b>" + localtime + "<BR>");
document.write("<b>Абсолютное время: </b>" + utctime + "</p>");
</SCRIPT>
</BODY>
</HTML>
```

Вот вы и получили документ HTML со сценарием JavaScript. Сохраните его с расширением .html или .htm.

Тестирование сценария

Для того, чтобы протестировать сценарий, необходимо просто загрузить полученную вэб-страницу в браузер. Откройте страницу в вашем браузере. Если вы все сделали правильно, то должны получить примерно следующее:

Текущее время

Текущее время: Tue Jul 26 13:41:19 UTC+0400 2005
Абсолютное время: Tue, 26 Jul 2005 09:41:19 UTC

Только время и дата будут ваши.

Наверняка вы заметили, что отображаемое время имеет не очень привлекательный вид. Чтобы отобразить значение красиво необходимо немного преобразовать программу сценария. Мы создали сценарий для вычисления всеобщего скоординированного времени.

```
<SCRIPT LANGUAGE="JavaScript">
now = new Date();
localtime=now.toString();
utctime=now.toGMTString();
document.write("<b>Текущее время: </b>" + localtime + "<BR>");
document.write("<b>Абсолютное время: </b>" + utctime + "</p>");
</SCRIPT>
```

Если помните, время отображалось не в очень привлекательном виде, вот так:

Текущее время

Текущее время: Tue Jul 26 13:41:19 UTC+0400 2005
Абсолютное время: Tue, 26 Jul 2005 09:41:19 UTC

Для того чтобы отобразить время красиво, придется немного преобразовать программу сценария. Отобразим время так, как это делает большой электронный будильник. Для этого необходимо воспользоваться некоторыми дополнительными средствами JavaScript и HTML.

Введем три новые переменные: для часов, минут и секунд. И в этом случае все сложные вычисления выполнят встроенные функции JavaScript. Код сценария будет таким:

```
1:   <HTML>
2:   <HEAD><TITLE>Отображение даты</TITLE>
3:   <BODY>
4:   <H1>Текущее время</H1>
5:   <p>
6:     <SCRIPT LANGUAGE="JavaScript">
7:     now = new Date();
8:     localtime=now.toString();
9:     utctime=now.toGMTString();
10:     hours=now.getHours();
11:     mins=now.getMinutes();
12:     secs=now.getSeconds();
13:     document.write("<b>Текущее время: </b>" + localtime + "<BR>");
14:     document.write("<b>Абсолютное время: </b>" + utctime + "</p>");
15:     document.write("<font size='+5'>");
16:     document.write(hours + ":" + mins + ":" + secs);
17:     document.write("</font>");
18:  </SCRIPT>
19:  </BODY>
20:  </HTML>
```

Обращаю внимание, что номера строк введены для удобства, и при написании сценария их писать**не нужно**.

Три новые переменные заданы в строках 10-12. В них сохраняются часы, минуты и секунды текущего времени.

В строке 15 добавляется дескриптор . Он используется для отображения значений шрифтом большого размера. В следующей строке (16) на экран выводится значение всех трех новых переменных, разделенных символом двоеточия (:). В строке 17 с помощью JavaScript вводится закрывающий тег .

После изменения сценария сохраните код вэб-документа в формате HTML. Откройте его в браузере. Если оставить браузер открытым и периодически нажимать кнопку ОБНОВИТЬ, то можно обновлять время в его окне. Протестируйте сценарий, если вы все ввели правильно, то должно получиться примерно следующее:

Текущее время

Текущее время: Tue Aug 2 17:11:21 UTC+0400 2005
Абсолютное время: Tue, 2 Aug 2005 13:11:21 UTC

17:11:21

Только время будет ваше.

Выявление и устранение ошибок

Вот мы и создали полезный сценарий JavaScript и даже успели его изменить. Теперь изменим сценарий таким образом, чтобы он содержал ошибку.

Вы можете удивиться: зачем добавлять ошибку в работающий сценарий? Ответ: рано или поздно вы все равно столкнетесь с ошибками в сценариях. Поскольку на простом примере их выявлять гораздо проще, познакомимся как ведет себя браузер при обнаружении ошибки в программах JavaScript.

Для создания ошибки, например измените оператор в строке 17, удалите закрывающую скобку. Тогда оператор будет выглядеть вот так:

document.write("";

Сохраните документ и обновите его в браузере. В зависимости от версии браузера вы получите либо сообщение об ошибке, либо сценарий просто не будет выполняться.

При отображении сообщения об ошибке вам легче утранить ее. Если сообщение не отображается, необходимо настроить браузер таким образом, чтобы он автоматически тестировал сценарии. Опять-таки, выполняемые операции зависят от используемого браузера.

В Netscape Navigator 4.5 и выше включено средство JavaScript Console, которое отображает сообщения об ошибках. Чтобы отобразить консоль JavaScript, в поле адреса браузера введите**javascript:**.

В Internet Explorer версии 4.0 и выше выберите Tools => Internet Options (Сервис => Свойства обозревателя). На вкладке Advanced (Дополнительно) снимите флажок опции Disable Script Debugging (Запретить отладку сценариев) и поставьте флажок Display a notification every script error (Выводить сообщение о каждой ошибке сценария).

В нашем случае сообщение **missing) after argument list**, правильно определяет источник ошибки. Но, будьте готовы к тому, что сообщение об ошибке не всегда правильно определяет ее причину. В любом случае, в окне консоли JavaScript указывается строка, в которой встречается ошибка.

Скрытие сценариев от старых браузеров

Поскольку старые версии браузеров не выполняют сценарии JavaScript и не понимают дескриптор <SCRIPT>, приходится всячески изгаляться, чтобы правильно отобразить вэб-

страницу. В большинстве случаев программный код сценария просто отображается в средней части страницы...

Чтобы избежать подобного поведения браузеров, используются символы комментариев. Они дают команду старым браузерам игнорировать код сценария. Новые браузеры достаточно "умны", чтобы понять, что программа сценария, определенная как комментарий, на самом деле должна выполняться.

Комментарии в HTML начинаются с <!-- и заканчиваются -->. В следующем примере приведен код простого сценария, скрытого комментариями:

```
1:  <SCRIPT LANGUAGE="JavaScript">
2:  <--
3:  document.write("Ваш браузер поддерживает JavaScript");
4:  // -->
5:  </SCRIPT>
```

Этот сценарий включает открывающий и закрывающий дескрипторы комментариев. Оператор // последний в комментарии. Он защищает комментарий HTML от определения его как ошибки в коде JavaScript.

Процедура скрытия сценария JavaScript не идеальна. Отдельные символы (подобные символу >) могут обозначить конец комментария раньше, чем это необходимо.

Практическая работа №27
Программирование объекта Window

Объект window

Класс объектов Window - это самый старший *класс* в иерархии объектов JavaScript. *Объект* window, относящийся к текущему окну (т.е. в котором выполняется *скрипт*), является объектом класса Window. *Класс объектов Frame* содержится в классе Window, т.е. каждый*фрейм* - это тоже *объект* класса Window.

О *фреймах* речь пойдет ниже, а пока вернемся к объекту window. *Объект* window создается только в момент открытия окна. Все остальные объекты, которые порождаются при загрузке страницы, есть свойства объекта window. Более того, все *глобальные переменные*, определенные в данном окне, тоже являются свойствами объекта window. Таким образом, у объекта window могут быть разные свойства при загрузке разных страниц. Кроме того, в разных браузерах свойства объектов и поведение объектов и браузера при*обработке событий* может быть различным. При программировании на JavaScript чаще всего используют следующие свойства, методы и события объекта window:

Таблица 4.1. Свойства, методы и события объектаwindow

Свойства	**Методы**	**События**
status	open()	Load
defaultStatus	close()	Unload
location	focus()	
history	blur()	Focus
navigator		Blur
	alert()	
document	confirm()	Resize
frames[]	prompt()	Error
opener	setTimeout()	
parent	setInterval()	
self	clearTimeout()	
top	clearInterval()	

Поскольку *объект* window является самым старшим, то в большинстве случаев при обращении к его свойствам и методам приставку "window." можно опускать (разумеется, в случае, если вы хотите обратиться к свойству или методу текущего окна, где работает *скрипт*; если же это другое окно, то необходимо указать его *идентификатор*). Так, например, можно писать *alert*('Привет') вместоwindow.*alert*('Привет'), или *location* вместо window.*location*. Исключениями из этого правила являются вызовы методовopen() и close(), у которых нужно указывать *имя окна*, с которым работаем (родительское в первом случае и дочернее во втором). Свойства *frames*[], self, *parent* и top будут рассмотрены в разделе, посвященном фреймам. Свойство opener будет рассмотрено при описании метода window.close().

Свойства объекта window

Поле статуса и свойство window.status

Поле статуса - это первое, что начали использовать авторы HTML-страниц из арсенала JavaScript. Калькуляторы, игры, математические вычисления и другие элементы выглядели слишком искусственно. На их фоне бегущая строка в поле статуса была изюминкой, которая могла действительно привлечь внимание пользователей к Web-узлу. Постепенно ее популярность сошла на нет. Бегущие строки стали редкостью, но программирование поля статуса встречается на многих Web-узлах.

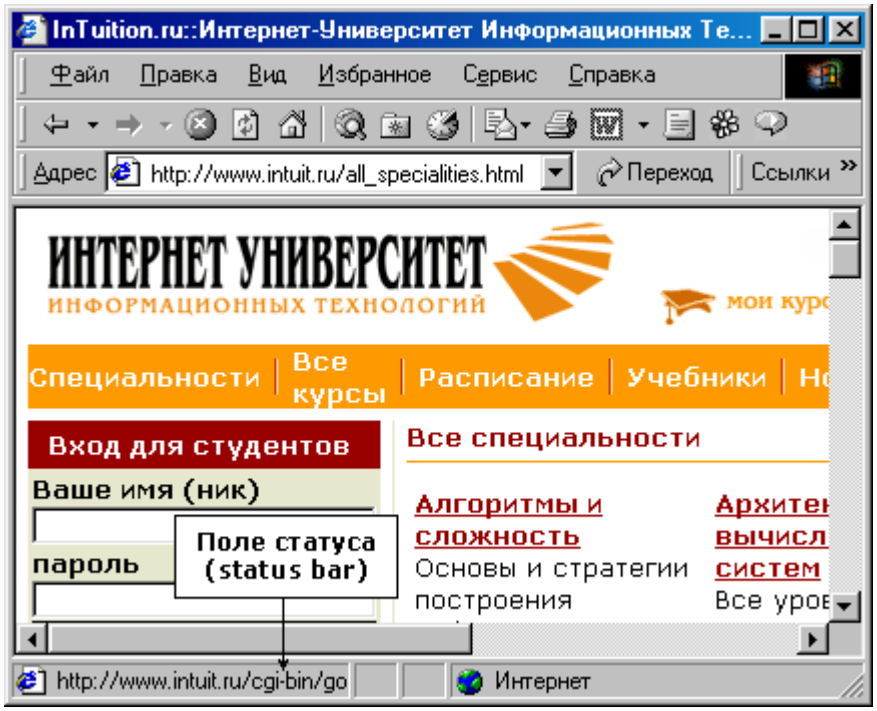


Рис. Поле статуса

Полем статуса (*status bar*) называют поле нижней части окна браузера сразу под областью отображения HTML-страницы. В поле статуса отображается информация о состоянии браузера (загрузка документа, загрузка графики, завершение загрузки, запуск *апплета* и т.п.). Программа на JavaScript имеет возможность работать с этим полем как с изменяемым свойством окна. При этом фактически с ним связаны два разных свойства:

- window.*status* - значение поля статуса;
- window.defaultStatus - значение поля статуса по умолчанию.

Значение свойства *status* можно изменить - и оно тут же будет отображено в поле статуса. Свойство defaultStatus тоже можно менять - и сразу по его изменении оно отображается в поле статуса.

Разница между этими двумя свойствами заключается в их поведении: если свойству *status* присвоить пустую строку:window.*status*="", то в поле статуса автоматически будет отображено значение defaultStatus. Обратного же не происходит: при присвоении пустой строки свойству defaultStatus оно и отобразится в поле статуса, независимо от значения свойства *status*. Следует отметить, что реакция браузеров на описываемые ниже действия со свойствами *status* и defaultStatus может быть разной в различных браузерах.

Программируем status

Свойство *status* связано с отображением сообщений о событиях, отличных от простой загрузки страницы. Например, в *Internet Explorer*при наведении указателя мыши на ссылку обработчик onMouseOver помещает в поле статуса значение URL, указанное в атрибуте *HREF*этой ссылки (при этом никак не меняя значения свойств *status* и defaultStatus). При попадании же курсора мыши на область, свободную от

ссылок, обработчик onMouseOut возвращает в поле статуса значение defaultStatus, при условии, что оно не есть пустая строка (опять же никак не меняя значений обоих свойств). Мы можем изменить это поведение, например, как в следующем примере:

```
<A onMouseOver="window.status='Мышь над ссылкой';return true;"
   onMouseOut="window.status='Мышь увели со ссылки';"
  HREF="http://site.com/">Наведите мышь на ссылку и следите за полем статуса</A>
```

Обратите внимание на оператор return true в конце обработчика событий onMouseOver. Он необходим для того, чтобы *отменить*действие по умолчанию (в данном случае - вывод URL в поле статуса), которое, в отсутствие этого оператора, браузер выполнил бы сразу после вывода нами своей строки в поле статуса, и пользователь не успел бы увидеть нашу строку. Аналогичный трюк отмены действия по умолчанию годится и для некоторых других событий (*onClick*, onKeyDown, onKeyPress, onMouseDown, onMouseUp, onSubmit,onReset), с той лишь разницей, что для перечисленных обработчиков отмена выполняется оператором return false.

Для обработчика onMouseOut такого способа отменить действие по умолчанию не существует (к сожалению). Но в данном конкретном случае это не требуется - как уже было сказано, при уводе курсора со ссылки в поле статуса восстанавливается значениеdefaultStatus только в случае, если это значение не есть пустая строка. Но в нашем случае (по умолчанию при загрузке страницы в IE) оно равно именно пустой строке. Поэтому, уводя курсор с нашей ссылки, мы продолжаем видеть в поле статуса строку "Мышь увели со ссылки". Ситуация изменится в следующем примере, когда мы предварительно зададим свое (непустое) значение defaultStatus.

Программируем defaultStatus

Свойство defaultStatus определяет текст, отображаемый в поле статуса, когда никаких событий не происходит. Дополним предыдущий пример изменением этого свойства в момент окончания загрузки документа, т.е. в обработчике onLoad:

```
<BODY onLoad="window.defaultStatus='Значение по умолчанию';">

<A onMouseOver="window.status='Мышь над ссылкой';return true;"
   onMouseOut="window.status='Мышь увели со ссылки'; alert('Ждем');"
  HREF="http://site.com/">Наведите мышь на ссылку и следите за полем статуса</A>

</BODY>
```

Сразу после загрузки документа в поле статуса будет "Значение по умолчанию". При наведении указателя мыши на ссылку в поле статуса появится надпись "Мышь над ссылкой", при этом URL ссылки (http://site.com/) в поле статуса не появится, т.к. мы подавили его вывод оператором return true.

При убирании указателя мыши со ссылки пользователь бы не успел увидеть строку "Мышь увели со ссылки", поскольку действие по умолчанию (вывод значения defaultStatus в поле статуса) не подавлено (и не может быть подавлено - у обработчика onMouseOutнет такой

возможности). Однако мы ввели оператор вывода *окна предупреждения alert*('Ждем') (он рассматривается ниже) - и теперь пользователь будет видеть в поле статуса строку "Мышь увели со ссылки" до тех пор, пока не нажмет OK в этом окне.

Поле адреса и свойство window.location

Поле адреса в браузере обычно располагается в верхней части окна и отображает URL загруженного документа. Если пользователь хочет вручную перейти к какой-либо странице (набрать ее URL), он делает это в поле адреса.

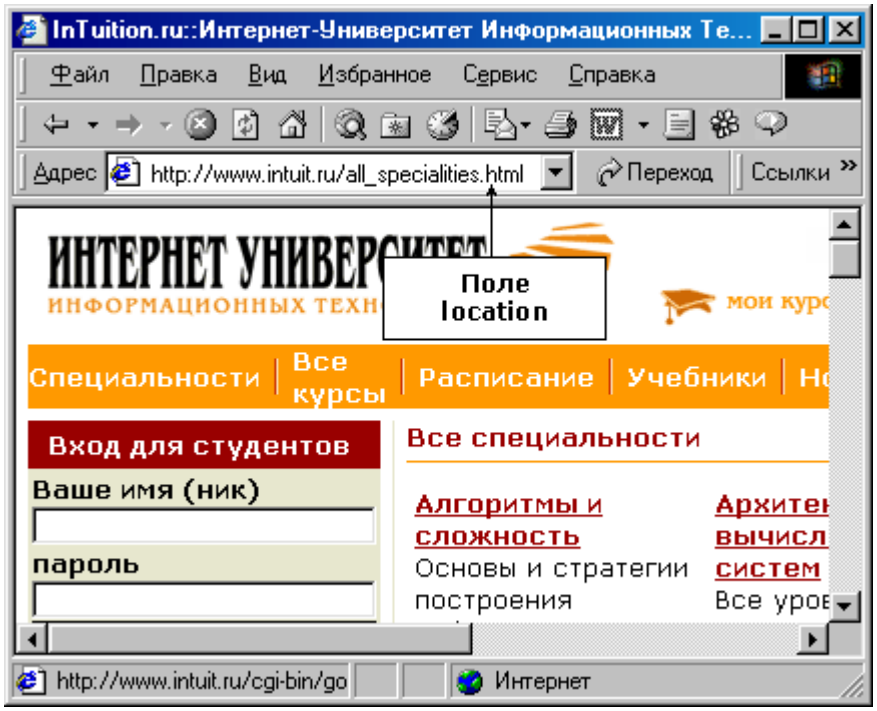


Рис. Поле адреса (location)

Свойство *location* объекта window само является объектом класса *Location*. Класс *Location*, в свою очередь, является*подклассом* класса URL, к которому относятся также объекты классов *Area* и Link. Объекты *Location* наследуют все свойства объектов URL, что позволяет получить доступ к любой части *схемы URL*. Подробнее о *классе объектов* URL мы расскажем в"Программируем гипертекстовые переходы" .

В целях совместимости с прежними версиями JavaScript, в языке поддерживается также свойство window.document.*location*, которое в настоящее время полностью дублирует свойство window.*location* со всеми его свойствами и методами. Рассмотрим теперь свойства и методы объекта window.*location* (событий, связанных с этим объектом, нет).

Свойства объекта location

Их проще продемонстрировать на примере. Предположим, что браузер отображает страницу, расположенную по адресу:

http://www.site.ru:80/dir/page.cgi?product=phone&id=3#mark

Тогда свойства объекта *location* примут следующие значения:

```
window.location.href     = "http://www.site.ru:80/dir/page.cgi?product=phone&id=3#mark"
window.location.protocol = "http:"
window.location.hostname = "www.site.ru"
window.location.port     = 80
window.location.host     = "www.site.ru:80"
window.location.pathname = "dir/page.cgi"
window.location.search   = "?product=phone&id=3"
window.location.hash     = "#mark"
```

Как уже говорилось в предыдущих лекциях, к свойствам объектов можно обращаться как с помощью *точечной нотации* (как выше), так и с помощью *скобочной нотации*, например: window.*location*['*host*'].

Методы объекта location

Методы объекта *location* предназначены для управления загрузкой и перезагрузкой страницы. Это управление заключается в том, что можно либо перезагрузить текущий документ (метод *reload*()), либо загрузить новый (метод *replace*()).

```
window.location.reload(true);
```

Метод *reload*() полностью моделирует поведение браузера при нажатии на кнопку *Reload* в панели инструментов. Если вызывать метод без аргумента или указать его равным true, то браузер проверит время последней модификации документа и загрузит его либо из *кеша*(если документ не был модифицирован), либо с сервера. Такое поведение соответствует простому нажатию кнопки *Reload* браузера (клавиши F5 в *Internet Explorer*). Если в качестве аргумента указать false, то браузер перезагрузит текущий документ с сервера, несмотря ни на что. Такое поведение соответствует одновременному нажатию клавиши Shift и кнопки браузера *Reload* (или Ctrl+F5 в*Internet Explorer*).

Используя объект *location*, перейти на новую страницу можно двумя способами:

```
window.location.href="http://www.newsite.ru/";
window.location.replace("http://www.newsite.ru/");
```

Разница между ними - в отображении этого действия в истории посещений страниц window.*history*. В первом случае в историю посещений добавится новый элемент, содержащий адрес " http://www.newsite.ru/ ", так что при желании можно будет нажать кнопку Back на панели браузера, чтобы вернуться к прежней странице. Во втором случае новый адрес " http://www.newsite.ru/ " заместит прежний в истории посещений, и вернуться к прежней странице нажатием кнопки Back уже будет невозможно.

История посещений (history)

История посещений страниц World Wide Web позволяет пользователю вернуться к странице, которую он просматривал ранее в данном окне браузера. История посещений в JavaScript трансформируется в объект window.*history*. Этот объект указывает на массив

URL-страниц, которые пользователь посещал и которые он может получить, выбрав из меню браузера режим Go. Методы объекта *history*позволяют загружать страницы, используя URL из этого массива.

Чтобы не возникло проблем с безопасностью браузера, путешествовать по *History* можно, только используя индекс. При этом URL, как текстовая строка, программисту недоступен. Чаще всего этот объект используют в примерах или страницах, на которые могут быть ссылки из нескольких разных страниц, предполагая, что можно вернутся к странице, из которой пример будет загружен:

```
<FORM><INPUT TYPE="button" VALUE="Назад" onClick="history.back()"></FORM>
```

Данный код отображает кнопку "Назад", нажав на которую, мы вернемся на предыдущую страницу. Аналогичным образом действует метод*history.forward*(), перенося нас на следующую посещенную страницу.

Существует также метод go(), имеющий целочисленный аргумент и позволяющий перескакивать на несколько шагов вперед или назад по истории посещений. Например, *history*.go(-3) перенесет нас на 3 шага назад в истории просмотра. При этом методы back() и*forward*() равносильны методу go() с аргументами -1 и 1, соответственно. Вызов *history*.go(0) приведет к перезагрузке текущей страницы.

Тип браузера (navigator)

Часто возникает задача настройки страницы на конкретную программу просмотра (браузер). При этом возможны два варианта: определение типа браузера на стороне сервера, либо на стороне клиента. Для последнего варианта в арсенале объектов JavaScript существует объект window.*navigator*. Важнейшие из свойств этого объекта перечислены ниже.

Таблица 4.2. Основные свойства объекта window.*navigator*

Свойство	**Описание**
userAgent	Основная информация о браузере. Передается серверу в HTTP-заголовке при открытии пользователем страниц
appName	Название браузера
appCodeName	Кодовое название браузера
appVersion	Данные о версии браузера и совместимости

Рассмотрим простой пример определения типа программы просмотра:

```
<FORM><INPUT TYPE=button VALUE="Тип навигатора"
onClick="alert(window.navigator.userAgent);"></FORM>
```

При нажатии на кнопку отображается *окно предупреждения*, содержащее значение свойства *navigator*.userAgent. Если это значение разобрать по компонентам, то может получиться, например, следующее:

```
navigator.userAgent   = "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; SLCC1)"
```

```
navigator.appName    = "Microsoft Internet Explorer"
navigator.appCodeName = "Mozilla"
navigator.appVersion  = "4.0 (compatible; MSIE 7.0; Windows NT 6.0; SLCC1)"
```

У объекта *navigator* есть еще несколько интересных с точки зрения программирования применений. Например, чтобы проверить, поддерживает ли браузер клиента язык Java, достаточно вызвать метод *navigator*.javaEnabled(), возвращающий значение true, если поддерживает, и false в противном случае.

Можно проверить, какие форматы графических файлов поддерживает браузер, воспользовавшись свойством *navigator*.mimeTypes (оно представляет собой массив всех типов *MIME*, которые поддерживаются данным браузером):

```
<SCRIPT>
if(navigator.mimeTypes['image/gif']!=null)
  document.write('Ваш браузер поддерживает GIF<BR>');
if(navigator.mimeTypes['image/tiff']==null)
  document.write('Ваш браузер не поддерживает TIFF');
</SCRIPT>
```

К сожалению, такая проверка не позволяет определить наличие возможности автоматической подгрузки графики.

Практическая работа №28
Методы объекта объекта Window

Методы объекта window

Что можно сделать с окном? Открыть (создать), закрыть (удалить), положить его поверх всех других открытых окон (передать фокус). Кроме того, можно управлять свойствами окна и свойствами подчиненных ему объектов. Сосредоточимся на простых и наиболее популярных *методах управления* окнами.

alert()

Метод *alert*() позволяет выдать *окно предупреждения*, имеющее единственную кнопку "OK":

```
<A HREF="javascript:window.alert('Внимание')">
Повторите запрос!</A>
```

Нужно лишь иметь в виду, что сообщения выводятся *системным шрифтом*, следовательно, для получения предупреждений на русском языке нужна локализованная версия ОС.

confirm()

Метод *confirm*() позволяет задать пользователю вопрос, на который тот может ответить либо положительно (нажав кнопку "OK"), либо отрицательно (нажав кнопку "Отмена" или "*Cancel*", либо просто закрыв окно запроса). В соответствии с действиями пользователя метод *confirm*() возвращает значение true либо false. Пример:

```
<FORM NAME=f>
<INPUT TYPE=button NAME=b VALUE="Нажмите эту кнопку"
onClick="if(window.confirm('Вы знаете JavaScript?'))
    document.f.b.value='Да. Спросить еще?';
else document.f.b.value='Нет. Спросить еще?';">
</FORM>
```

Все ограничения для сообщений на русском языке, которые были описаны для метода *alert*(), справедливы и для метода *confirm*().

prompt()

Метод prompt() позволяет принять от пользователя строку текста. Синтаксис его таков:

prompt("Строка вопроса","Строка ответа по умолчанию")

Когда пользователь введет свой ответ (либо оставит неизменным ответ по умолчанию) и нажмет кнопку OK, метод prompt() возвратит полученную строчку в качестве значения, которое можно далее присвоить любой переменной и потом разбирать ее в JavaScript-программе.

```
<FORM NAME=f>
<INPUT TYPE=button VALUE="Открыть окно ввода"
onClick="document.f.e.value=
    window.prompt('Введите сообщение','Сюда');">
<INPUT SIZE=30 NAME=e>
</FORM>
```

window.open()

Метод ***open()*** предназначен для создания новых *окон* . В общем случае его синтаксис выглядит следующим образом:

myWin = window.open("URL","имя_окна","параметр=значение,параметр=значение,...", заменить);

Первый аргумент задает адрес страницы, загружаемой в новое окно (можно оставить пустую строку, тогда окно останется пустым). Второй аргумент задает *имя окна*, которое можно будет использовать в атрибуте *TARGET* контейнеров <A> и <FORM>. В качестве значений допустимы также зарезервированные имена _blank, _parent, _self, _top, смысл которых такой же, как у аналогичных значений атрибута *TARGET*. Если имя_окна совпадает с именем уже существующего окна (или *фрейма*), то новое окно не создается, а все последующие манипуляции с переменной myWin будут применяться к этому окну (или *фрейму*).

Третий аргумент есть **не содержащая пробелов** строка, представляющая собой список параметров и их значений, перечисленных через запятую. Указание каждого из параметров необязательно, однако значения по умолчанию могут зависеть от браузера, поэтому всегда указывайте явно те параметры, на которые рассчитываете. Возможные параметры перечислены в таблице 4.3. Вместо значенийyes и no можно использовать 1 и 0. Последний

аргумент " заменить " является необязательным, принимает значения true иfalse и означает: следует ли новый URL добавить в *history* в качестве нового элемента или заменить им последний элемент*history*.

Метод window.open() возвращает ссылку на вновь открытое окно, т.е. объект класса Window. Его можно присвоить переменной (что мы и сделали выше), с тем чтобы потом можно было управлять открытым окном (писать в него, читать из него, передавать и убирать фокус, закрывать).

Таблица 4.3. Параметры метода window.open()

Параметр	**Значения**	**Описание**
width	число	Ширина окна в пикселах (не менее 100)
height	число	Высота окна в пикселах (не менее 100)
left	число	Расстояние от левого края экрана до левой границы окна в пикселах
top	число	Расстояние от верхнего края экрана до *верхней границы* окна в пикселах
directories	yes / no	Наличие у окна панели папок (*Netscape Navigator*)
location	yes / no	Наличие у окна поля адреса
menubar	yes / no	Наличие у окна панели меню
resizable	yes / no	Сможет ли пользователь менять размер окна
scrollbars	yes / no	Наличие у окна *полос прокрутки*
status	yes / no	Наличие у окна поля статуса
toolbar	yes / no	Наличие у окна панели инструментов

Приведем два примера открытия нового окна:

```
<FORM>
<INPUT TYPE=button VALUE="Простое окно"
 onClick="window.open('', 'test1',
  'directories=no,height=200,location=no,'+
  'menubar=no,resizable=no,scrollbars=no,'+
  'status=no,toolbar=no,width=200');">

<INPUT TYPE=button VALUE="Сложное окно"
 onClick="window.open('', 'test2',
  'directories=yes,height=200,location=yes,'+
  'menubar=yes,resizable=yes,scrollbars=yes,'+
  'status=yes,toolbar=yes,width=200');">
</FORM>
```

При нажатии кнопки "Простое окно" получаем окно со следующими параметрами:

- directories=no - окно без панели папок
- height=200 - высота 200 px
- *location*=no - поле адреса отсутствует
- menubar=no - без меню
- *resizable*=no - размер окна изменять нельзя
- scrollbars=no - *полосы прокрутки* отсутствуют

- *status*=no - статусная строка отсутствует
- *toolbar*=no - системные кнопки браузера отсутствуют
- width=200 - ширина 200 px

При нажатии кнопки "Сложное окно" получаем окно, где:

- directories=yes - окно с панелью папок
- height=200 - высота 200 px
- *location*=yes - поле адреса есть
- menubar=yes - меню есть
- *resizable*=yes - размер изменять можно
- scrollbars=yes - есть *полосы прокрутки*
- *status*=yes - статусная строка есть
- *toolbar*=yes - системные кнопки браузера есть
- width=200 - ширина 200 px

window.close()

Метод ***close()*** позволяет закрыть окно. Чаще всего возникает вопрос, какое из окон, собственно, следует закрыть. Если необходимо закрыть текущее, то:

```
window.close();
self.close();
```

Если мы открыли окно с помощью метода window.open(), то из скрипта, работающего в новом окне, сослаться на окно-родитель можно с помощью window.opener (обратите внимание, здесь window ссылается на объект нового, созданного окна, т.к. оно использовано в скрипте, работающем в новом окне). Поэтому, если необходимо закрыть родительское окно, т.е. окно, из которого было открыто текущее, то:

```
window.opener.close();
```

Если необходимо закрыть произвольное окно, то тогда сначала нужно получить его идентификатор:

```
id=window.open();
...
id.close();
```

Как видно из последнего примера, закрывают окно не по имени (значение атрибута *TARGET* тут ни при чем), а используют указатель на объект.

Методы focus() и blur()

Метод *focus*() применяется для передачи фокуса в окно, с которым он использовался. Передача фокуса полезна как при открытии окна, так и при его закрытии, не говоря уже о случаях, когда нужно выбирать окна. Рассмотрим пример.

Открываем окно и, не закрывая его, снова откроем окно с таким же именем, но с другим текстом. Новое окно не появилось поверх основного окна, так как фокус ему не был передан. Теперь повторим открытие окна, но уже с передачей фокуса:

```
<HTML>
<HEAD>
<SCRIPT>
function myfocus(a)
{
 myWin = window.open('','example','width=300,height=200');
 // открываем окно и заводим переменную с указателем на него.
 // Если окно с именем 'example' существует, то новое окно не создается,
 // а открывается поток для записи в имеющееся окно с именем 'example'

 if(a==1)
 {
  myWin.document.open(); //открываем поток ввода в уже созданное окно
  myWin.document.write('<H1>Открыли окно в первый раз'); //Пишем в этот поток
 }

 if(a==2)
 {
  myWin.document.open();
  myWin.document.write('<H1>Открыли окно во второй раз');
 }

 if(a==3)
 {
  myWin.focus(); // передаем фокус, а затем выполняем те же действия,
          // что и в предыдущем случае
  myWin.document.open();
  myWin.document.write('<H1>Открыли окно в третий раз');
 }

 myWin.document.write('</H1>');
 myWin.document.close();
}
</SCRIPT>
</HEAD>
<BODY>
<a href="javascript:myfocus(1);">Откроем окно и напишем в него что-то</a>,
<BR><BR>
<a href="javascript:myfocus(2);">напишем в него же что-то другое, но фокус не передадим</a>,
<BR><BR>
<a href="javascript:myfocus(3);">опять что-то напишем в него, но сперва передав ему фокус</a>.
</BODY>
</HTML>
```

4.1. Передача фокуса в новое окно

Поскольку мы пишем содержание нового окна из окна старого (родителя), то в качестве указателя на объект используем значение переменной myWin.

Чтобы увести фокус из определенного окна myWin, необходимо применить метод myWin.*blur*(). Например, чтобы увести фокус с текущего окна, где выполняется скрипт, нужно вызвать window.*blur*(). Эффект будет тот же, как если бы пользователь сам свернул окно нажатием кнопки  в правом верхнем углу окна.

Метод setTimeout()

Метод *setTimeout*() используется для создания нового потока вычислений, исполнение которого откладывается на время (в миллисекундах), указанное вторым аргументом:

idt = setTimeout("JavaScript_код",Time);

Типичное применение этой функции - организация периодического изменения свойств объектов. Например, можно запустить часы в поле формы:

```
<HTML><HEAD><SCRIPT>
var Chasy_idut=false;

function myclock()
{
  if(Chasy_idut)
  {
    d = new Date();
    document.f.c.value =
    d.getHours()+':'+
    d.getMinutes()+':'+
    d.getSeconds();
  }
 setTimeout("myclock();",500);
}

function FlipFlag()
{
  Chasy_idut = !Chasy_idut;
  document.f.b.value = (Chasy_idut)?
   'Остановить' : 'Запустить';
}
</SCRIPT></HEAD>
<BODY onLoad="myclock();">
<FORM NAME=f>
Текущее время:<INPUT NAME=c size=8>
<INPUT TYPE=button name=b VALUE="Запустить"
onClick="FlipFlag();">
</FORM></BODY></HTML>
```

4.2. Часы с использованием setTimeout()

Обратите внимание, что поток порождается (т.е. вызывается *setTimeout*()) всегда, даже в том случае, когда мы остановили показ часов. Если бы он создавался только при значении переменной Chasy_idut = true, то часы бы просто не запустились, так как в самом начале исполнения скрипта мы установили var Chasy_idut = false. Но даже если бы мы установили в начале var Chasy_idut = true, то часы бы запустились при загрузке страницы, а после остановки поток бы исчез, и при последующем нажатии кнопки "Запустить" часы продолжали бы стоять.

Метод clearTimeout()

Метод clearTimeout() позволяет уничтожить поток, вызванный методом *setTimeout()* . Очевидно, что его применение позволяет более эффективно распределять ресурсы вычислительной установки. Для того чтобы использовать этот метод в примере с часами, нам нужно модифицировать функции и форму:

```
<HTML><HEAD><SCRIPT>
var Chasy_idut=false;
var potok;

function StartClock()
{
  d = new Date();
  document.f.c.value =
   d.getHours()+':'+
   d.getMinutes()+':'+
   d.getSeconds();
  potok = setTimeout('StartClock();',500);
  Chasy_idut=true;
}

function StopClock()
{
  clearTimeout(potok);
  Chasy_idut=false;
}
</SCRIPT></HEAD><BODY>
<FORM NAME=f>
Текущее время:<INPUT NAME=c size=8>
<INPUT TYPE=button VALUE="Запустить" onClick="if(!Chasy_idut) StartClock();">
<INPUT TYPE=button VALUE="Остановить" onClick="if(Chasy_idut) StopClock();">
</FORM></BODY></HTML>
```

4.3. Часы с использованием setTimeout() и clearTimeout()

В данном примере для остановки часов используется метод clearTimeout(). При этом, чтобы не порождалось множество потоков, проверяется значение указателя на объект потока.

Методы setInterval() и clearInterval()

В предыдущих примерах для того, чтобы поток запускался снова и снова, мы помещали в функцию в качестве последнего оператора *вызов метода setTimeout*(). Однако в JavaScript для этих целей имеются специальные методы. Метод*setInterval*("код_JavaScript",time) выполняет код_JavaScript с периодом раз в time миллисекунд. *Возвращаемое значение* - ссылка на созданный поток. Чтобы остановить поток, необходимо вызвать метод clearInterval(поток).

События объекта window

Остановимся вкратце на событиях, связанных с объектом window. Обработчики этих событий обычно помещают как *атрибут* контейнера<BODY>.

- *Load* - событие происходит в момент, когда загрузка документа в данном окне полностью закончилась. Если текущим окном является*фрейм*, то событие *Load* его объекта window происходит, когда в данном фрейме загрузка документа закончилась, независимо от состояния загрузки документов в других *фреймах*. Использовать обработчик данного события можно, например, следующим образом:
  <BODY onLoad="alert('Документ полностью загружен.');">
- Unload - событие происходит в момент выгрузки страницы из окна. Например, когда пользователь закрывает окно, либо переходит с данной Web-страницы на другую, кликнув ссылку или набрав адрес в адресной строке, либо при изменении адреса страницы (свойствawindow.*location*) скриптом. Например, при уходе пользователя с нашей страницы мы можем позаботиться о его удобстве и закрыть открытое ранее нашим скриптом окно:
  <BODY onUnload="myWin.close();">
- Error - событие происходит при возникновении ошибки в процессе загрузки страницы. Если это событие произошло, можно, например, вывести сообщение пользователю с помощью *alert*() или попытаться перезагрузить страницу с помощьюwindow.*location.reload*(). В следующем примере мы назначаем обработчиком события Error функцию ff(), которая будет выдавать сообщение. В тексте программы мы допустили ошибку: слово *Alert* написано с заглавной буквы (помните, что в JavaScript это недопустимо?). Поэтому при открытии этого примера возникнет ошибка и пользователь получит об этом "дружественное" сообщение.
  - <SCRIPT>
  - function ff()
  - { alert('Произошла ошибка. Свяжитесь с Web-мастером.'); }
  -
  - window.onerror = ff;
  -
  - Alert('Привет');
    </SCRIPT>
- *Focus* - событие происходит в момент, когда окну передается фокус. Например, когда пользователь "раскрывает" свернутое ранее окно, либо (в Windows) выбирает это окно браузера с помощью Alt+*Tab* среди окон других приложений. Это событие происходит также при программной передаче фокуса данному окну путем вызова метода window.*focus*(). Пример использования:
  <BODY onFocus="alert('Спасибо, что снова вернулись!');">

- *Blur* - событие, противоположное предыдущему, происходит в момент, когда данное окно теряет фокус. Это может произойти в результате действий пользователя либо *программными средствами* - вызовом метода window.*blur*().
- *Resize* - событие происходит при изменении размеров окна пользователем либо сценарием.

Переменные как свойства окна

Глобальные переменные на самом деле являются свойствами объекта window. В следующем примере мы открываем окно с идентификатором *wid*, заводим в нем *глобальную переменную* t и затем пользуемся ею в окне-родителе, ссылаясь на нее как *wid*.t:

```
<HTML><HEAD>
<SCRIPT>
wid = window.open('','width=750,height=100,status=yes');
wid.document.open(); R = wid.document.write;
R('<HTML><HEAD><SCRIPT>var t;<\/SCRIPT></HEAD>');
R('<BODY><H1>Новое окно</H1></BODY></HTML>');
wid.document.close();
</SCRIPT>
</HEAD>
<BODY>
<A HREF="javascript:
wid.t=window.prompt('Новое состояние:','');
wid.document.write(wid.t); wid.focus(); void(0);"
>Изменим значение переменной t в новом окне</A>
</BODY></HTML>
```

4.4. Изменение переменной открытого окна

Обратите внимание на нюанс: внутри скрипта мы написали <\/SCRIPT>. Комбинация " \/ " выдает на выходе " / ". Сделали мы это для того, чтобы *браузер* (точнее, его HTML-*парсер*) не воспринял бы </SCRIPT> как завершающий тэг нашего (внешнего) скрипта. Подробнее этот аспект обсуждался во вводной лекции. Также обратите внимание на *алиас* (*синоним*) R, который мы дали методу*wid*.document.write, чтобы иметь возможность кратко вызывать его как R(...).

Аналогичным образом (с приставкой *wid*, указывающей на *объект* окна) можно обращаться ко всем элементам, находящимся в открытом нами окне, например, к формам. В качестве примера рассмотрим изменение поля ввода в окне-потомке из окна-предка. Создадим дочернее окно с помощью функции okno(), в нем создадим форму, а затем обратимся к полю этой формы из окна-предка:

```
<HTML>
<HEAD>
<SCRIPT>
var wid; // Объявляем глобальную переменную
function okno()
{
  wid = window.open('','okoshko','width=500,height=200');
  wid.document.open(); R = wid.document.write;
  R('<HTML><BODY><H1>Меняем текст в окне-потомке:</H1>');
```

```
 R('<FORM NAME=f><INPUT SIZE=40 NAME=t VALUE=Текст>');
 R('</FORM></BODY></HTML>');
 wid.document.close();
}
</SCRIPT>
</HEAD>
<BODY>
<INPUT TYPE=button VALUE="Открыть окно примера" onClick="okno()">
<INPUT TYPE=button VALUE="Написать текущее время в поле ввода"
     onClick="window.wid.document.f.t.value=new Date();
          window.wid.focus();">
</BODY>
</HTML>
```

4.5. Изменение поля статуса в открытом окне

Открывая окно-*потомок*, мы поместили в переменную *wid указатель* на окно: *wid*=window.open(...);. Теперь мы можем использовать *wid* как *идентификатор объекта* класса Window. *Вызов метода* window.*wid.focus*() в нашем случае обязателен, поскольку при нажатии на кнопку "Написать *текущее время* в *поле ввода*" происходит передача фокуса в родительское окно (которое может заслонять вновь открытое окно, так что изменения, происходящие в окне-потомке, не будут видны пользователю). Для того, чтобы увидеть изменения, мы должны передать фокус в окно-*потомок*.

Переменная wid должна быть *глобальной*, т.е. определена за пределами каких-либо функций (как сделано в нашем примере). В этом случае она становится *свойством* объекта window, поэтому мы обращаемся к ней в обработчике *onClick* посредством window.*wid*. Если бы мы поместили ее внутри описания функции okno(), написав var *wid* = window.open(...), то мы не смогли бы к ней обратиться из обработчика события *onClick*, находящегося вне функции okno().

Практическая работа №29
Программирование объекта Document

Объект document

Объект document является важнейшим свойством объекта window (т.е. полностью к нему нужно обращаться как window.document). Все элементы HTML-разметки, присутствующие на web-странице - текст, абзацы, гиперссылки, картинки, списки, таблицы, формы и т.д. - являются свойствами объекта document. Можно сказать, что технология *DHTML* (*Dynamic HTML*), т.е. динамическое изменение содержимого web-страницы, заключается именно в работе со свойствами, методами и событиями объекта document (не считая работы с окнами и *фреймами*).

Таблица Свойства, методы и события объекта document

Свойства	**Методы**	**События**
URL domain title lastModified	open() close() write()	Load Unload Click

referrer cookie linkColor alinkColor vlinkColor	writeln() getSelection() getElementById() getElementsByName() getElementsByTagName()	DblClick MouseDown MouseUp KeyDown KeyUp KeyPress

Помимо перечисленных в этой таблице свойств, *объект* document имеет свойства, являющиеся коллекциями (форм, картинок, ссылок и т.п.); *таблица* 3.1 предыдущей лекции содержит их описание. Кроме того, можно формировать требуемые коллекции "на лету" с помощью указанных выше методов. Так, document.getElementsByTagName('P') есть коллекция всех HTML-элементов (точнее, соответствующих им объектов) вида <P>, т.е. абзацев. Аналогично, document.getElementsByName('important') выдаст *коллекцию (объектов*) HTML-элементов любых типов, у которых был задан *атрибут* NAME="important". Наконец,document.*getElementById*('id5') выдаст тот HTML-элемент (если их несколько, то первый), у которого был задан *атрибут*ID="id5".

С одним методом мы уже часто работали: document.write() - он пишет в текущий HTML-документ. Его модификацияdocument.writeln() делает то же самое, но дополнительно добавляет в конце *символ новой строки*; это удобно, если потом требуется читать сгенерированный HTML-документ глазами. Если *запись* идет в HTML-документ нового окна, открытого с помощьюwindow.open(), то перед записью в него нужно открыть *поток* на *запись* с помощью метода document.open(), а по окончании записи закрыть *поток* методом document.close(). После выполнения последнего действия произойдет событие *Load* (и вызовется соответствующий обработчик события onLoad) у документа, а затем у окна.

События объекта document аналогичны одноименным событиям объекта window (только у document они происходят раньше), либо их смысл понятен из их названия, поэтому мы не будем детально их разбирать.

Остановимся вкратце на свойствах объекта document. Свойства linkColor, alinkColor и vlinkColor задают цвет гиперссылок - непосещенных, активных и посещенных, соответственно. Свойство URL хранит *адрес* текущего документа (т.е. *строковый литерал*, равный window.*location.href*, если страница состоит из единственного документа, а не является набором *фреймов*). Свойство*domain* выдает *домен* (оно аналогично window.*location.hostname*). Свойство title выдает *заголовок страницы* (указанный в контейнере <TITLE>), lastModified указывает на дату и время последней модификации файла, в котором содержится данный HTML-документ (без учета времени модификации *внешних файлов* - стилевых, скриптов и т.п.). Свойство referrer выдает *адрес* страницы, с которой *пользователь* пришел на данную web-страницу, кликнув по гиперссылке. Наконец, свойству *cookie* посвящен *целый* раздел в"Программируем "за кадром"" .

Практическая работа №30
Фреймы

Фреймы (Frames)

Фреймы - это несколько видоизмененные окна. Отличаются они от обычных окон тем, что размещаются внутри них. У *фрейма* не может быть ни панели инструментов, ни *меню*, как в обычном окне. В качестве поля статуса *фрейм* использует *поле* статуса окна, в котором он размещен. Существует и ряд других отличий.

Если окно имеет фреймовую структуру (т.е. вместо контейнера <BODY> в нем присутствует *контейнер* <*FRAMESET*> со вложенными в него контейнерами <*FRAME*> и быть может другими контейнерами <*FRAMESET*>), то *объект* window соответствует внешнему контейнеру <*FRAMESET*>, а с каждым вложенным контейнером <*FRAME*> ассоциирован свой собственный *объект* класса Window.

Каждому окну или *фрейму* создатель страницы может дать *имя* - с помощью атрибута NAME контейнера *FRAME*, либо вторым аргументом метода window.open(). Используется оно в качестве значения атрибута *TARGET* контейнеров A и FORM, чтобы открыть ссылку или отобразить результаты работы формы в определенном окне или фрейме. Есть несколько зарезервированных имен окон:_self (имя текущего окна или *фрейма*, где исполняется *скрипт*), _blank (новое окно), _parent (окно-родитель для данного*фрейма*), _top (самый старший предок данного *фрейма*, т.е. окно браузера, частью которого является данный *фрейм*). *Иерархияфреймов*, обсуждаемая ниже, как раз и задает, какие окна или фреймы являются родителями для других *фреймов*.

У каждого объекта класса Window, будь то окно или *фрейм*, есть также *ссылка* на соответствующий *объект*. Как мы знаем, ссылкой на*объект* текущего окна, в котором исполняется *скрипт*, является window ; кроме того, на него же ссылается свойство self объектаwindow (а также свойство window объекта window - есть и такое!). Ссылку на *объект* окна, открываемого методом window.open(), выдает сам этот метод. *Ссылка* на *объект-фрейм* совпадает с его именем, заданным с помощью атрибута NAME контейнера *FRAME*. Наконец, у объектов-*фреймов* есть специальные свойства, дающие ссылки на родительский *фрейм* (window.*parent*) и на окно браузера, частью которого является данный *фрейм* (window.top).

Таким образом, для того, чтобы правильно обращаться к нужным фреймам, нам нужно знать лишь их *иерархию*, т.е. взаимное подчинение (какой *фрейм* для какого является родителем). Это мы сейчас и обсудим.

Иерархия и именование фреймов

Рассмотрим сначала простой пример. Разделим экран на две вертикальные колонки:

```
<HTML>
<HEAD>
<TITLE>Левый и правый</TITLE>
</HEAD>
 <FRAMESET COLS="50%,*">
   <FRAME NAME=leftframe SRC=left.htm>
   <FRAME NAME=rightframe SRC=right.htm>
 </FRAMESET>
```

</HTML>

4.6. Два фрейма

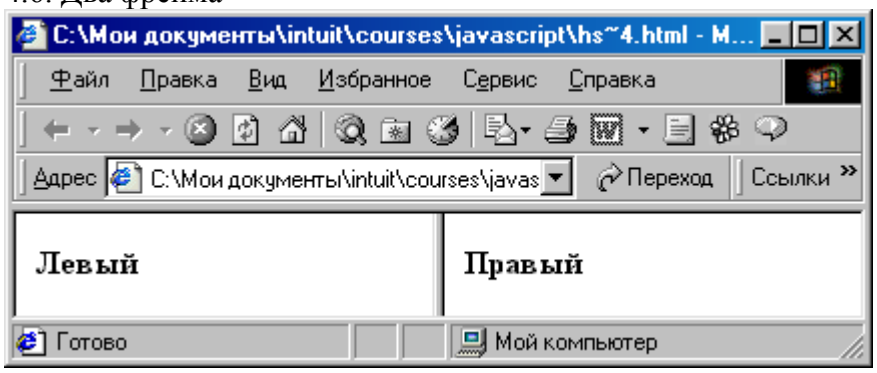


Рис. Окно с двумя вертикальными фреймами

Иерархия *фреймов* здесь получается следующая:

- window
  - leftframe
  - rightframe

Из основного окна (из скрипта, который можно было поместить в контейнер <HEAD>) обратиться к левому *фрейму* можно с помощьюwindow.leftframe, к правому - window.rightframe. Из каждого *фрейма* обратиться к основному окну можно как window.*parent*либо window.top (что в данном случае равносильно) или даже просто *parent* и top (так как приставку window можно опускать). Наконец, из левого *фрейма* обратиться к правому *фрейму* можно как *parent*.rightframe или top.rightframe.

Усложним пример: разобьем правый *фрейм* на два по горизонтали:

```
<HTML>
<HEAD>
<TITLE>Левый, верх и низ</TITLE>
</HEAD>
 <FRAMESET COLS="50%,*">
  <FRAME NAME=leftframe SRC=left.htm>

  <FRAMESET ROWS="50%,*">
    <FRAME NAME=topframe SRC=top.htm>
    <FRAME NAME=botframe SRC=bottom.htm>
  </FRAMESET>

 </FRAMESET>
</HTML>
```

4.7. Три фрейма

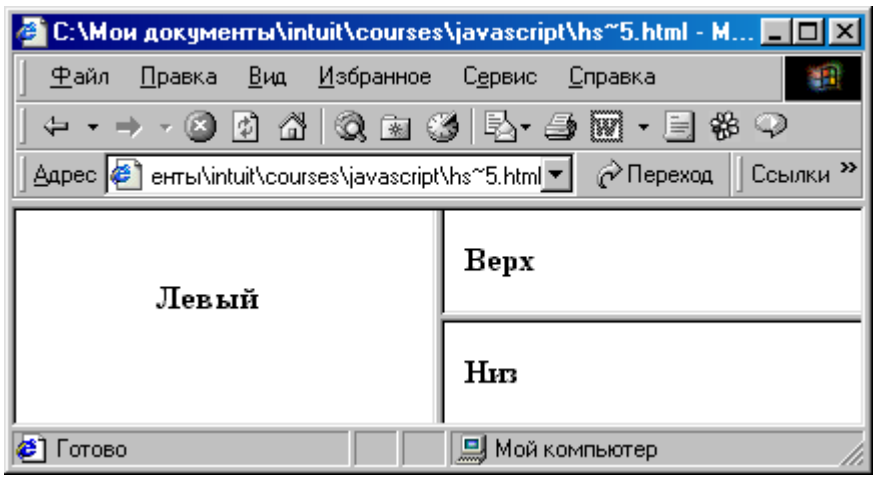


Рис. Правый фрейм разбит на два по горизонтали

Фрейма с именем rightframe теперь не существует. Более того, все три *фрейма* непосредственно подчинены главному окну, т.е. иерархия выглядит следующим образом:

- window
  - leftframe
  - topframe
  - botframe

Следовательно, мы можем поместить в контейнер <HEAD> следующий скрипт, устанавливающий *цвет фона* для всех трех *фреймов*: (открыть)

```
<SCRIPT>
window.onload=f;
function f()
{
 window.leftframe.document.bgColor='blue';
 window.topframe.document.bgColor='red';
 window.botframe.document.bgColor='green';
}
</SCRIPT>
```

Для того чтобы *фрейм* rightframe все же появился в иерархии и ему подчинялись два правых *фрейма*, нужно свести оба наших примера в один. Это значит, что во *фрейм* rightframe мы должны загрузить отдельный фреймовый документ.

Основной документ

```
<HTML>
<HEAD>
</HEAD>
 <FRAMESET COLS="50%,*">
  <FRAME NAME=leftframe SRC=left.htm>
```

Документ в правом фрейме (right.*htm*)

```
<HTML>
<HEAD>
</HEAD>
 <FRAMESET ROWS="50%,*">
   <FRAME NAME=topframe SRC=top.htm>
```

```
  <FRAME NAME=rightframe  <FRAME NAME=botframe
SRC=right.htm>                SRC=bottom.htm>
 </FRAMESET>                   </FRAMESET>
</HTML>                       </HTML>
```

В этом случае иерархия *фреймов* будет выглядеть иначе:

- window
  - leftframe
  - rightframe
    - topframe
    - botframe

Теперь чтобы из главного окна обратиться ко всем трем фреймам и установить в них те же *цвета фона*, следует писать:

```
window.leftframe.document.bgColor='blue';
 window.rightframe.topframe.document.bgColor='red';
 window.rightframe.botframe.document.bgColor='green';
```

Таким образом, визуально на Web-странице мы получили тот же результат, что и с тремя *фреймами*, подчиненными одному старшему окну (см. пример 4.7). Однако этот вариант более гибкий: он позволяет работать независимо с *фреймом* rightframe в отдельном файле.

Коллекция фреймов

Выше мы обращались к *фрейму* по его имени. Однако, если имя не известно (или не задано), либо если нужно обратиться ко всем дочерним фреймам по очереди, то более удобным будет обращение через *коллекцию фреймов frames*[], которая является свойством объекта window.

В качестве иллюстрации предположим, что в примере из двух *фреймов* (пример 4.6) правый *фрейм* содержит несколько изображений, и нам требуется поменять адрес (значение атрибута SRC) третьего изображения с помощью скрипта, находящегося в левом фрейме. Правый *фрейм* - второй, значит, его номер 1; третье изображение имеет номер 2. Поэтому, это можно сделать следующими способами:

```
top.frames[1].document.images[2].src = 'pic.gif';
 top.frames['rightframe'].document.images[2].src = 'pic.gif';
 top.frames.rightframe.document.images[2].src = 'pic.gif';
 top.rightframe.document.images[2].src = 'pic.gif';
```

Передача данных во фрейм

Обычной задачей при разработке типового Web-узла является загрузка результатов исполнения *CGI-скрипта* во *фрейм*, отличный от*фрейма*, в котором вводятся данные для этого скрипта. Если путь загрузки результатов фиксированный, то можно просто использовать атрибут *TARGET* формы. Сложнее, если результат работы должен быть загружен в разные фреймы (например, в зависимости от выбранной кнопки).

Применим полученные нами знания для решения этой задачи. Сначала заготовим следующие файлы. Основной файл, например,index.*htm*, содержит левый *фрейм*, в котором будет находиться форма, и правый *фрейм*, разбитый на два подфрейма (верхний и нижний). Файл left.*htm* содержит форму, в которой пользователю предоставляется возможность выбрать верхний или нижний *фрейм*и нажать кнопку "Загрузить". Файл right.*htm* содержит простой текст; он будет загружаться в верхний или нижний *фрейм*, в зависимости от действий пользователя.

Основной файл с тремя *фреймами*

```
<HTML>
<HEAD>
<TITLE>Три фрейма</TITLE>
</HEAD>
 <FRAMESET COLS="50%,*">
  <FRAME NAME=leftframe
       SRC=left.htm>
  <FRAMESET
ROWS="50%,*">
   <FRAME NAME=topframe
SRC="">
   <FRAME NAME=botframe
SRC="">
  </FRAMESET>
 </FRAMESET>
</HTML>
```

Файл с формой left.*htm* в левом фрейме

```
<HTML>
<HEAD>
<SCRIPT SRC="loadframe.js"></SCRIPT>
</HEAD>
<BODY>
 <FORM METHOD=post
ACTION=right.htm
  NAME=f onSubmit="return load();">
  <SELECT NAME=s>
    <OPTION>верхний</OPTION>
    <OPTION>нижний</OPTION>
  </SELECT>
<INPUT TYPE=submit
VALUE="Загрузить">
 </FORM>
</BODY>
</HTML>
```

Файл right.*htm*

```
<HTML>
<BODY>
Этот документ
мы загружаем
при выборе
фрейма
из списка

</BODY>
</HTML>
```

Для того, чтобы пример заработал, остается в файле loadframe.js описать функцию *load*(). Функция должна делать так, чтобы в зависимости от выбора пользователем значения селектора "верхний" или "нижний" файл right.*htm* загружался бы либо в правый верхний, либо в правый нижний *фрейм*. С этой целью в файле left.*htm* у формы не был указан целевой *фрейм* (атрибут *TARGET*).

Нашу задачу динамического выбора *фрейма* можно решать по-разному. Более изящный способ - переназначать "на лету" свойство*target*, с него мы и начнем (открыть).

```
function load()
{
if(document.f.s.selectedIndex==0)
  {
   document.f.target = "topframe";
   top.frames[2].document.open();
   top.frames[2].document.close();
  }
else
  {
   document.f.target = "botframe";
```

```
  top.frames[1].document.open();
  top.frames[1].document.close();
 }
return true;
}
```

4.8. Файл loadframe.js: переназначение target на лету

Функция *load*() всегда возвращает true, а поскольку она вызывается из обработчика события onSubmit, это означает, что всегда будет происходить отправка формы (событие Submit), т.е. загрузка страницы right.*htm*, указанной в атрибуте *ACTION* данной формы. Обратите внимание также на следующие строки в функции *load*():

```
top.frames[1].document.open();
  top.frames[1].document.close();
```

Смысл их таков: когда пользователь выбирает значение верхний или нижний в форме, то файл right.*htm* загружается в соответствующий *фрейм*, а оставшийся *фрейм* открывается на запись (методом ...document.open(), при этом всё его содержимое очищается) и закрывается (методом ...document.close()), тем самым *фрейм* остаётся пустым (без текста).

Теперь рассмотрим второй подход - открытие окна с именем, совпадающим с именем *фрейма* topframe или botframe. Его идея состоит в том, что при попытке открыть окно с именем существующего окна новое окно не открывается, а используется уже открытое.*Фрейм* - это тоже окно, поэтому на него данное правило распространяется. Функция, реализующая такое поведение, приведена ниже (открыть):

```
function load()
{
 if(document.f.s.selectedIndex==0)
 {
  window.open("right.htm","topframe");
  top.frames[2].document.open();
  top.frames[2].document.close();
 }
 else
 {
  window.open("right.htm","botframe");
  top.frames[1].document.open();
  top.frames[1].document.close();
 }
return false;
}
```

4.9. Файл loadframe.js: использование window.open()

В этом подходе функция *load*() всегда возвращает false. Это необходимо, чтобы отменить *отправку данных* формы: ведь после того, как мы вызвали window.open(), в *отправке данных* формы, т.е. загрузке файла right.*htm*, уже нет надобности.

Практическая работа №31

Встроенный класс Date

Создаёт экземпляр объекта **Date**, представляющего собой момент времени. Объекты даты Date основываются на значении количества миллисекунд, прошедших с 1 января 1970 года в часовом поясе UTC.

Синтаксис

```
new Date();
new Date(value);
new Date(dateString);
new Date(year, month[, day[, hour[, minute[, second[, millisecond]]]]]);
```

Обратите внимание: объекты Date могут быть созданы только путём вызова функции Date в качестве конструктора: обычный вызов функции (то есть, без использования оператора new) вернёт строку вместо объекта Date; в отличие от других объектных типов JavaScript, объекты Date не имеют литерального синтаксиса.

Параметры

Обратите внимание: если функция Date вызывается в качестве конструктора с более, чем одним аргументом, значения, большие логического диапазона (например, 13 в качестве номера месяца или 70 для значения минут) «переметнутся» на соседние значения. Например, вызов new Date(2013, 13, 1) эквивалентенвызову new Date(2014, 1, 1), оба создадут дату 2014-02-01 (нумерация месяцев начинается с нуля). Тоже самое действует и для других значений: вызов new Date(2013, 2, 1, 0, 70)эквивалентен вызову new Date(2013, 2, 1, 1, 10) — оба вызова создадут дату 2013-03-01T01:10:00.

Обратите внимание: если функция Date вызывается в качестве конструктора с более чем одним аргументом, то указанные аргументы интерпретируются как локальное время. Если аргументы указывают время в UTC, используйтеnew Date(Date.UTC(...)) с теми же аргументами.

value

Целое значение, представляющее количество миллисекунд, прошедших с 1 января 1970 00:00:00 по UTC (эпохи Unix).

dateString

Строковое значение, представляющее дату. Строка должна быть в одном из форматов, распознаваемых методом Date.parse() (совместимые с IETF RFC 2822 временные метки [на английском, на русском], а также версия ISO8601 [на английском, на русском]).

year

Целое значение, представляющее год. Значения с 0 по 99 отображаются на года с 1900 по 1999. Смотрите пример ниже.

month

Целое значение, представляющее месяц, начинается с 0 для января и кончается 11 для декабря.

day

Необязательный параметр. Целое значение, представляющее день месяца.

hour

Необязательный параметр. Целое значение, представляющее часы дня.

minute

Необязательный параметр. Целое значение, представляющее минуты времени.

second

Необязательный параметр. Целое значение, представляющее секунды времени.

millisecond

Необязательный параметр. Целое значение, представляющее миллисекунды времени.

Описание

- Если никаких аргументов передано не было, конструктор создаёт объект Date для текущих даты и времени, согласно системным настройкам.
- Если передано как минимум два аргумента, отсутствующие аргументы устанавливаются либо в 1 (если опущен день), либо в 0 (все остальные значения).
- Дата в JavaScript измеряется в миллисекундах, прошедших с полуночи 1 января 1970 года по UTC. День содержит 86 400 000 миллисекунд. Диапазон дат объекта Dateварьируется от -100 000 000 до 100 000 000 дней относительно 1 января 1970 года по UTC.
- Объект Date обеспечивает универсальное поведение на всех платформах. Значение времени может передаваться между системами для представления одинакового момента во времени и, если оно используется для создания локального объекта даты, будет отражать местный эквивалент времени.
- Объект Date поддерживает несколько методов для работы с UTC (всемирным координированным временем), наряду с методами работы с местным временем. UTC, также известное как среднее время по Гринвичу (GMT), ссылается на время, установленное Всемирным стандартом времени. Местное время — это время на компьютере, на котором выполняется JavaScript.
- Вызов объекта Date в качестве функции (то есть, без использования операторanew) вернёт строку, представляющую текущие дату и время.

Свойства

Справку по свойствам, доступным на экземплярах Date, смотрите в разделе Свойства экземпляров Date.

Date.prototype

Позволяет добавлять свойства к объекту Date.

Date.length

Значение свойства Date.length равно 7. Это количество аргументов, обрабатываемых конструктором.

Свойства, унаследованные из Function:
arity, caller, constructor, length, name

Методы

Справку по методам, доступным на экземплярах Date, смотрите в разделе Методы экземпляров Date.

Date.now()

Возвращает числовое значение, соответствующее текущему времени — количество миллисекунд, прошедших с 1 января 1970 года 00:00:00 по UTC.

Date.parse()

Разбирает строковое представление даты и возвращает количество миллисекунд с 1 января 1970 года 00:00:00 по местному времени.

Date.UTC()

Принимает те же самые параметры, что и самый длиный вариант конструктора (то есть, от 2 до 7) и возвращает количество миллисекунд, прошедших с 1 января 1970 года 00:00:00 по UTC.

Методы, унаследованные из Function:
apply, call, toSource, toString

Экземпляры объекта Date

Все экземпляры объекта Date наследуются от Date.prototype. Объект прототипа конструктора Date может быть изменён, чтобы затронуть все экземпляры объектаDate.

Методы

Получения значения

Date.prototype.getDate()

Возвращает день месяца (1-31) указанной даты по местному времени.

Date.prototype.getDay()

Возвращает день недели (0-6) указанной даты по местному времени.

Date.prototype.getFullYear()

Возвращает год (4 цифры для 4-х значного года) указанной даты по местному времени.

Date.prototype.getHours()

Возвращает часы (0-23) указанной даты по местному времени.

Date.prototype.getMilliseconds()

Возвращает миллисекунды (0-999) указанной даты по местному времени.

Date.prototype.getMinutes()

Возвращает минуты (0-59) указанной даты по местному времени.

Date.prototype.getMonth()

Возвращает месяц (0-11) указанной даты по местному времени.

Date.prototype.getSeconds()

Возвращает секунды (0-59) указанной даты по местному времени.

Date.prototype.getTime()

Возвращает числовое значение указанной даты как количество миллисекунд, прошедших с 1 января 1970 года 00:00:00 по UTC (отрицательное значение для даты до этого момента).

Date.prototype.getTimezoneOffset()

Возвращает смещение часового пояса в минутах для текущей локали.

Date.prototype.getUTCDate()

Возвращает день месяца (1-31) указанной даты по всемирному координированному времени.

Date.prototype.getUTCDay()

Возвращает день недели (0-6) указанной даты по всемирному координированному времени.

Date.prototype.getUTCFullYear()

Возвращает год (4 цифры для 4-х значного года) указанной даты по всемирному координированному времени.

Date.prototype.getUTCHours()

Возвращает часы (0-23) указанной даты по всемирному координированному времени.

Date.prototype.getUTCMilliseconds()

Возвращает миллисекунды (0-999) указанной даты по всемирному координированному времени.

Date.prototype.getUTCMinutes()

Возвращает минуты (0-59) указанной даты по всемирному координированному времени.

Date.prototype.getUTCMonth()

Возвращает месяц (0-11) указанной даты по всемирному координированному времени.

Date.prototype.getUTCSeconds()

Возвращает секунды (0-59) указанной даты по всемирному координированному времени.

Date.prototype.getYear()

Возвращает год (обычно 2-3 цифры) указанной даты по всемирному координированному времени. Вместо него используйте метод getFullYear().

Установки значения

Date.prototype.setDate()

Устанавливает день месяца указанной даты по местному времени.

Date.prototype.setFullYear()

Устанавливает полный год (4 цифры для 4-х значного года) указанной даты по местному времени.

Date.prototype.setHours()

Устанавливает часы указанной даты по местному времени.

Date.prototype.setMilliseconds()

Устанавливает миллисекунды указанной даты по местному времени.

Date.prototype.setMinutes()

Устанавливает минуты указанной даты по местному времени.

Date.prototype.setMonth()

Устанавливает месяц указанной даты по местному времени.

Date.prototype.setSeconds()

Устанавливает секунды указанной даты по местному времени.

Date.prototype.setTime()

Устанавливает объект Date во время, представляемое количеством миллисекунд, прошедших с 1 января 1970 года 00:00:00 по UTC (отрицательное значение устанавливает даты до этого момента).

Date.prototype.setUTCDate()

Устанавливает день месяца указанной даты по всемирному координированному времени.

Date.prototype.setUTCFullYear()

Устанавливает полный год (4 цифры для 4-х значного года) указанной даты по всемирному координированному времени.

Date.prototype.setUTCHours()

Устанавливает часы указанной даты по всемирному координированному времени.

Date.prototype.setUTCMilliseconds()

Устанавливает миллисекунды указанной даты по всемирному координированному времени.

Date.prototype.setUTCMinutes()

Устанавливает минуты указанной даты по всемирному координированному времени.

Date.prototype.setUTCMonth()

Устанавливает месяц указанной даты по всемирному координированному времени.

Date.prototype.setUTCSeconds()

Устанавливает секунды указанной даты по всемирному координированному времени.

Date.prototype.setYear()

Устанавливает год (обычно 2-3 цифры) указанной даты по всемирному координированному времени. Вместо него используйте метод setFullYear().

Получения преобразованного значения

Date.prototype.toDateString()

Возвращает часть, содержащую только дату объекта Date в качестве человеко-читаемой строки.

Date.prototype.toISOString()

Преобразует дату в строку, следуя расширенному формату ISO 8601.

Date.prototype.toJSON()

Возвращает строку, представляющую объект Date, используя метод toISOString(). Предназначен для использования методом JSON.stringify().

Date.prototype.toGMTString()

Возвращает строку, представляющую объект Date, на основе часового пояса GMT (всемирное врямя). Вместо него используйте метод toUTCString().

Date.prototype.toLocaleDateString()

Возвращает строку с датой, чьё представление зависит от системных настроек локали.

Date.prototype.toLocaleFormat()

Преобразует дату в строку, используя строку форматирования.

Date.prototype.toLocaleString()

Возвращает строку, чьё представление зависит от настроек локали. Переопределяет метод Object.prototype.toLocaleString().

Date.prototype.toLocaleTimeString()

Возвращает строку со временем, чьё представление зависит от системных настроек локали.

Date.prototype.toSource()

Возвращает строковое представление исходного кода эквивалентного объекта Date; вы можете использовать это значение для создания нового объекта. Переопределяет метод Object.prototype.toSource().

Date.prototype.toString()

Возвращает строковое представление указанного объекта Date. Переопределяет метод Object.prototype.toString().

Date.prototype.toTimeString()

Возвращает часть, содержащую только время объекта Date в качестве человеко-читаемой строки.

Date.prototype.toUTCString()

Преобразует дату в строку, используя часовой пояс UTC.

Date.prototype.valueOf()

Возвращает примитивное значение объекта Date. Переопределяет методObject.prototype.valueOf().

Методы, унаследованные из Object:

__defineGetter__, __defineSetter__, hasOwnProperty, isPrototypeOf,__lookupGetter__, __lookupSetter__, __noSuchMethod__,propertyIsEnumerable, unwatch, watch

Примеры

Пример: несколько способов создания объекта Date

Следующие примеры показывают несколько способов создания дат в JavaScript:

```
var today = new Date();
var birthday = new Date('December 17, 1995 03:24:00');
var birthday = new Date('1995-12-17T03:24:00');
var birthday = new Date(1995, 11, 17);
var birthday = new Date(1995, 11, 17, 3, 24, 0);
```

Пример: двухциферный год отображается на 1900 - 1999 года

Для того, чтобы создать и получить даты между 0 и 99 годом, нужно использоваться методы Date.prototype.setFullYear() и Date.prototype.getFullYear().

```
var date = new Date(98, 1); // Sun Feb 01 1998 00:00:00 GMT+0000 (GMT)
```

```
// Устаревший метод, 98 отображается на 1998 год
date.setYear(98);          // Sun Feb 01 1998 00:00:00 GMT+0000 (GMT)

date.setFullYear(98);      // Sat Feb 01 0098 00:00:00 GMT+0000 (BST)
```

Пример: вычисление затраченного времени

Следующие примеры показывают, как определить разницу во времени между двумя датами в JavaScript:

```
// Используя объекты Date
var start = Date.now();

// Событие, для которого замеряется время, происходит тут:
doSomethingForALongTime();
var end = Date.now();
var elapsed = end - start; // затраченное время в миллисекундах
// Используя встроенные методы
var start = new Date();

// Событие, для которого замеряется время, происходит тут:
doSomethingForALongTime();
var end = new Date();
var elapsed = end.getTime() - start.getTime(); // затраченное время в миллисекундах
// Проверяет функцию и возвращает её возвращаемое значение
function printElapsedTime(fTest) {
  var nStartTime = Date.now(),
      vReturn = fTest(),
      nEndTime = Date.now();

  console.log('Затраченное время: ' + String(nEndTime - nStartTime) + ' миллисекунд');
  return vReturn;
}

yourFunctionReturn = printElapsedTime(yourFunction);
```

Практическая работа №32

Создание интерактивных элементов web-страниц

Цель: изучить методы создания интерактивных элементов web-страниц

Создание интерактивных компонентов Web-страниц

При создании Web-страниц с помощью программ Adobe Photoshop или Adobe ImageReady следует принимать во внимание конкретные возможности каждого из этих приложений:

- Photoshop обеспечивает подготовку статических изображений для Web с возможностью их деления на отдельные области, связанные гиперссылками с целевыми объектами;
- ImageReady кроме всех возможностей, предоставляемых Photoshop, включает также инструменты для создания динамических элементов, таких как анимации и интерактивные кнопки.

С помощью ImageReady можно создать целую анимированную Web-страницу, использующую комбинацию текстов, рисунков и графических объектов. При этом

предварительно следует продумать очередность их появления и перемещения в пределах страницы, которая должна иметь стандартный размер, чтобы полностью отображаться в окне обозревателя. Однако чаще всего ImageReady используют для создания таких интерактивных компонентов Web-страницы, как графические карты либо анимированные кнопки.

С этой целью в ImageReady добавлены специальные палитры для Web-дизайна: Rollover, Image Map и Slice, объединенные в одном окне с палитрой Animation (Анимация). Вывести их на передний план можно щелчком на соответствующей вкладке окна Animation (Анимация), либо выполнив одну из команд меню Window (Окно): Show Rollover, Show Slice, Show Image Map.

Под интерактивным эффектом (rollover) понимают такой эффект, при котором изображение принимает различный вид в зависимости от действий пользователя (например, наведение мыши или щелчок на определенной области Web-страницы). Каждое состояние определяется набором параметров палитры Layers (Слои), включая расположение слоев, их стили и параметры форматирования. В качестве состояния может быть использована анимация, либо можно создать такое изображение, при наведении мыши на один из участков которого изменяется вид другой части изображения (secondary rollovers).

Сохраняя изображение для использования в качестве элемента Web-страницы, можно одновременно сгенерировать HTML-файл, который будет содержать информацию для обозревателя о том, как воспроизводить элементы страницы при загрузке. Этот файл включает ссылки на изображения (в формате GIF, PNG или JPEG), HTML-текст, гиперссылки и код JavaScript для создания интерактивных эффектов (rollover effects). Хотя для большинства эффектов можно выполнить предварительный просмотр непосредственно в программах Photoshop или ImageReady, но зависимость демонстрируемых Web-страниц от операционной системы, типа обозревателя и системы отображения цвета требует выполнения просмотра в каждом конкретном обозревателе.

ImageReady поддерживает создание интерактивных объектов (rollovers), добавляя код JavaScript в результирующий HTML-файл, обеспечивающий смену состояния объекта при наведении на него указателя мыши. Для создания интерактивных объектов служит палитра Rollover, совмещенная с палитрой Animation (Анимация). На ней отображаются возможные состояния объекта, каждому из которых может соответствовать свое изображение или даже целая анимация. При сохранении такого объекта, как элемента Web-страницы каждое состояние сохраняется в отдельном файле, в название которого добавлен тип состояния.

Первое из состояний, отображенное на палитре Rollover, всегда состояние Normal (Обычное).

Остальные состояния определяют, каким действием пользователя будет вызываться соответствующий вид интерактивного объекта. Чтобы создать новое состояние, следует выбрать команду New State (Новое состояние) из меню палитры Rollover либо щелкнуть на одноименной кнопке этой палитры. Появившееся новое изображение соответствует следующему по порядку состоянию, но по виду идентично исходному и подлежит редактированию с помощью палитрыLayers (Слои). Изменить тип состояния можно с помощью раскрывающегося меню у имени состояния, содержащего следующие значения:

- Over (Наведение) — появление на интерактивном объекте указателя мыши при ненажатой левой кнопке;
- Down (Нажатие) — нажатие левой кнопки мыши при наведенном ее указателе на интерактивный объект (сохраняется, пока кнопка нажата);
- Click (Щелчок) — щелчок левой кнопки мыши на интерактивном объекте состояние сохраняется до следующего действия пользователя на объекте);

- Out (Наружу) — определяет состояние, когда указатель мыши сходит с области карты изображения (обычно для этой цели используется состояние Normal);
- Up (Отпускание) — определяет состояние, когда пользователь отпускает нажатую клавишу мыши на интерактивном объекте (обычно для этой цели используется состояние Over);
- Custom (Пользовательский) — определяет новое состояние, для которого нужно создать код JavaScript и добавить его к HTML-файлу Web-страницы, чтобы это действие могло быть реализовано;
- None (Нет) — используется, чтобы заданное изображение могло быть в дальнейшем использовано как одно из состояний интерактивного объекта (это состояние не отображается на Web-странице).

Раскрывающееся меню Rollover states (Интерактивные состояния) включает только те состояния, которые еще не применялись для данного объекта (исключения составляют состояния None иCustom, которые могут быть использованы неоднократно).

При работе со слоями изменение порядка слоев в состоянии Normal вызывает их перемещение во всех остальных состояниях. Однако перемещение отдельного слоя в других состояниях rolloversохраняет его первоначальное положение во всех остальных состояниях.

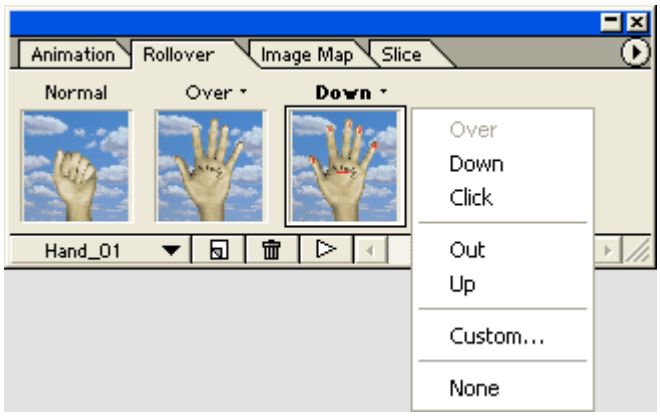


Рис. 3.15. Палитра Rollover с разными состояниями объекта

Так как палитры Animation и Rollover совмещены в одном окне, то легко можно создать анимацию, связанную с определенным состоянием. Предварительно следует выделить это состояние, затем перейти на палитру Animation (Анимация) и щелкнуть на кнопке New Frame (Новый кадр). Дальнейшее создание анимации выполняется способами, описанными ранее.

Анимация в состоянии Normal воспроизводится при первоначальной загрузке страницы, а остальные варианты анимации — при возникновении соответствующего события.

При использовании команд палитры Layers (Слои) для создания эффектов состояния интерактивных объектов (rollover) изменения действуют только на текущее состояние.

Чтобы изменения в слое отразились во всех состояниях интерактивных объектов (rollover), на палитре Layers (Слои) выделяем слой, содержащий элемент, который должен отображаться во всех состояниях, а на палитре Rollover из меню палитры выбираем одну из следующих команд:

Match Layer Across States (Подогнать слой под состояния) — назначает атрибуты выделенного слоя в текущем состоянии ко всем состояниям интерактивного объекта;

Match Layer Across All Rollovers (Подогнать слой под все интерактивные объекты) — назначает атрибуты выделенного слоя в текущем состоянии ко всем состояниям во всех интерактивных объектах данного изображения.

При копировании состояния интерактивного объекта (rollover) и вставке его в одно из состояний текущего объекта или объекта другого изображения слои вставляемого состояния заменят слои места назначения. Возможно также взаимное копирование и вставка кадров с палитры Animationили состояний с палитры Rollover как новых состояний или кадров анимации. При этом используется специальный внутренний буфер обмена, доступный только для этих команд, что не удаляет информацию из основного буфера обмена программы ImageReady.

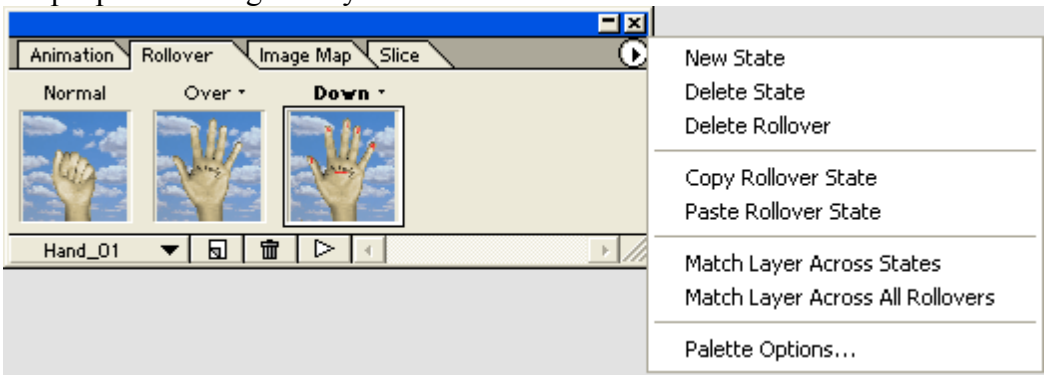


Рис. 3.16. Меню палитры Rollover

Чтобы скопировать состояние интерактивного объекта (rollover), следует выделить требуемое состояние и выполнить команду Copy Rollover State (Копировать состояние) из меню палитрыRollover. Затем выделяют состояние в текущем или другом интерактивном объекте, которое следует заменить, и выполняют команду Paste Rollover State (Вставить состояние) из меню палитры Rollover.

Возможно удаление отдельных состояний интерактивных объектов (rollover) либо всех состояний сразу. Чтобы удалить отдельное состояние, его следует выделить, а затем щелкнуть на кнопкеTrash (Корзина) палитры Rollover либо выбрать команду Delete State (Удалить состояние) из меню палитры Rollover. Чтобы удалить все состояния интерактивного объекта (rollover), выполняют команду Delete Rollover (Удалить интерактивный объект) из меню палитры Rollover.

Предварительный просмотр интерактивных объектов возможен непосредственно в окне документа ImageReady при переключении в режим rollover preview (например, с помощью кнопки Play(Воспроизвести) палитры Rollover). Этот просмотр выполняется с помощью обозревателя Internet Explorer 5.0 для Windows. Просмотр с помощью другого обозревателя выполняется щелчком на кнопке Preview in Default Browser (Просмотр в обозревателе по умолчанию) панели инструментов.

Для выхода из режима предварительного просмотра щелкают на кнопке Play (Воспроизвести) еще раз либо выбирают любой другой инструмент палитры инструментов.

Следует отметить, что в ImageReady инструмент Type (Текст) позволяет вводить текст прямо на изображение или графический объект в окне документа, а не в отдельное окно, как при использовании аналогичного инструмента Photoshop. Дальнейшее редактирование атрибутов текста возможно при его выделении и выборе команд контекстного меню, выполняющих их изменение.

Практическая работа №33
Работа с формами

Контейнер FORM

Если рассматривать *программирование* на JavaScript в исторической перспективе, то первыми объектами, для которых были разработаны методы и свойства, стали поля форм. Обычно *контейнер* FORM и поля форм именованы:

```
<FORM NAME=fname METHOD=get>
<INPUT NAME=iname SIZE=30 MAXLENGTH=30>
</FORM>
```

Поэтому в программах на JavaScript к ним обращаются по имени:

```
document.fname.iname.value="Текст";
```

Того же эффекта можно достичь, используя *коллекции* форм и элементов, обращаясь к форме и к элементу либо по индексу, либо по имени:

```
document.forms[0].elements[0].value="Текст";
document.forms['fname'].elements['iname'].value="Текст";
```

Рассмотрим подробнее *объект* Form, который соответствует контейнеру FORM. Его свойства, методы и события используются для задания реакции на действия пользователя, например, изменения значений полей или нажатие кнопок.

Свойства, методы и события объекта Form

Свойства	**Методы**	**События**
length	reset()	Reset
action	submit()	Submit
method		
target		
encoding		
elements[]		

Свойства объекта Form

Свойство action

Свойство *action* отвечает за вызов *CGI-скрипта*. В нем указывается URL этого скрипта. Но там, где можно указать URL, можно указать и его схему javascript:, например:

```
<FORM METHOD=post ACTION="javascript: alert('Работает!');">
<INPUT TYPE=submit VALUE="Продемонстрировать JavaScript в ACTION">
</FORM>
```

Обратите внимание на тот факт, что в контейнере FORM указан атрибут *METHOD*. В данном случае это сделано для того, чтобы к URL, заданному в атрибуте *ACTION*, не дописывался символ " ?". Дело в том, что *методом доступа* по умолчанию является метод GET. В этом методе при обращении к ресурсу из формы создается элемент URL под названием *search*. Этот элемент предваряется символом "?", который дописывается в конец URL скрипта. В нашем случае это привело бы к неправильной работе JavaScript-кода, поскольку конструкция вида

```
alert('Строка');?
```

провоцирует ошибку JavaScript. Метод *POST* передает данные формы скрипту в теле HTTP-сообщения, поэтому символ " ?" не добавляется к URL, и ошибка не генерируется. При этом применение void(0) отменяет перезагрузку документа, и браузер не генерирует событие *Submit*, т.е. не обращается к серверу при нажатии на кнопку, как это было бы при стандартной обработке формы.

Свойство method

Свойство *method* определяет *метод доступа* к ресурсам HTTP-сервера из программы-браузера. В зависимости от того, как автор HTML-страницы собирается получать и обрабатывать данные из формы, он может выбрать тот или иной *метод доступа*. На практике чаще всего используются методы GET и *POST.*

JavaScript-программа может изменить значение этого свойства. В предыдущем разделе *метод доступа* в форме был указан явно. Теперь мы его переопределим в момент исполнения программы:

```
<FORM NAME=f ACTION="javascript: alert('Работает!');">
<SCRIPT>
document.write('По умолчанию установлен метод: '+document.f.method+'.<BR>');
</SCRIPT>
<INPUT TYPE=button onClick="document.f.method='post'" VALUE="Сменить метод на POST">
<INPUT TYPE=button onClick="document.f.method='get'" VALUE="Сменить метод на GET"><BR>
<INPUT TYPE=submit VALUE="JavaScript в ACTION">
</FORM>
```

5.1. Изменение метода формы (GET и POST) скриптом

По умолчанию установлен метод GET.

В данном примере стоит обратить внимание на два момента:

1. Прежде чем открывать *окно предупреждения*, следует нажать кнопку "Метод *POST*". Если этого не сделать, то появится сообщение об ошибке JavaScript. Здесь все выглядит достаточно логично. Формирование URL происходит при генерации события *submit*, а вызов скрипта - после того, как событие сгенерировано. Поэтому вставить *переопределение метода* в обработчик события нельзя, так как к этому моменту будет уже сгенерирован URL, который, в свою очередь, будет JavaScript-программой с символом " ?" на конце.*Переопределение метода* должно быть выполнено раньше, чем произойдет событие *Submit.*
2. В *тело документа* через контейнер SCRIPT встроен JavaScript-код, который сообщает *метод доступа*, установленный в форме по умолчанию. Этот контейнер расположен сразу за контейнером FORM. Ставить его перед контейнером FORM нельзя, так как в момент получения *интерпретатором* управления объект FORM не будет создан, и, следовательно, работать с его свойствами не представляется возможным.

Никаких других особенностей свойство *method* не имеет. В данном свойстве можно указать и другие *методы доступа*, отличные от GETи *POST*, но это требует дополнительной настройки сервера.

Свойство target

Свойство *target* определяет *имя окна*, в которое следует загружать результат обращения к *CGI-скрипту*. При этом всегда есть альтернативы: можно использовать значение этого свойства внутри JavaScript-программ для указания окна или фрейма, куда требуется загружать результат работы *CGI-скрипта*, а можно получить идентификатор окна или задействовать встроенный массив *frames*[0] и свойства окна opener, top и *parent*. Кроме того, для загрузки *внешнего файла* в некоторое окно или *фрейм* можно также применить метод window.open(). Все эти варианты будут продемонстрированы в разделе "Передача данных во *фрейм*".

Свойство encoding

Свойство *encoding* объекта Form (а также атрибут enctype контейнера FORM) задает, каким образом данные из формы должны быть закодированы перед их отправкой на сервер. Возможные значения:

Значения свойства *encoding* объекта Form

Значение	**Описание**
application/x-www-form-urlencoded	Это значение по умолчанию. Означает, что в данных, передаваемых на сервер, пробелы заменяются на "+", а*специальные символы* заменяются на их 16-ричное *ASCII* значение, например, буква Щ заменяется на %D0%A9.
text/plain	Пробелы заменяются на "+", но *специальные символы* не кодируются (передаются как есть).
multipart/form-data	Никакие символы не кодируются (они передаются как есть). Данное значение необходимо указывать, если в форме имеются элементы отправки файлов: <INPUT TYPE=file>.

Коллекция elements[]

При генерации встроенного в документ объекта Form браузер создает и связанный с ним массив (*коллекцию*) полей формы*elements*[]. Обычно к полям обращаются по имени, но можно обращаться и по *индексу массива* полей формы:

```
<FORM NAME=f>
<INPUT NAME=e SIZE=40>
<BR><INPUT TYPE=button VALUE="Ввести текст по имени элемента"
  onClick="document.f.e.value='Текст введен по имени элемента';">
<BR><INPUT TYPE=button VALUE="Ввести текст по индексу элемента"
  onClick="document.f.elements[0].value='Текст введен по индексу элемента';">
<BR><INPUT TYPE=reset VALUE="Очистить">
</FORM>
```

Индексирование полей в массиве начинается с нуля. Общее число полей в форме f доступно двумя способами: как свойство массивадocument.f.*elements.length* и как свойство объекта формы: document.f.*length*.

Методы объекта Form
Метод submit()

Метод *submit*() позволяет проинициировать передачу введенных в форму данных на сервер:

```
<FORM NAME=f ACTION="http://www.intuit.ru/rating_students/">
Ваше имя пользователя на intuit:<INPUT NAME=query>
</FORM>
<A HREF="javascript:document.f.submit();">Посмотреть рейтинг</A>
```

Как видите, кнопки отправки (*submit*) у формы нет, но нажав на ссылку, мы выполняем *отправку данных* на сервер. Обычно при такой "скрытой" *отправке данных* на сервер браузеры, в *целях безопасности*, запрашивают подтверждение, действительно ли пользователь желает отправить данные. *Отправка данных* путем вызова метода *submit*() имеет отличия от нажатия пользователем кнопки INPUTтипа TYPE=*submit* ; их мы рассмотрим в конце лекции.

Метод reset()

Метод *reset*() (не путать с обработчиком события onReset, рассматриваемым ниже) позволяет восстановить значения полей формы, заданные по умолчанию. Другими словами, *вызов метода reset*() равносилен нажатию на кнопку INPUT типа TYPE=*reset*, но при этом саму эту кнопку создавать не требуется.

```
<FORM NAME=f>
<INPUT VALUE="Значение по умолчанию" SIZE=30>
<INPUT TYPE=button VALUE="Изменим текст в поле ввода"
     onClick="document.f.elements[0].value='Изменили текст';">
</FORM>
<A HREF="javascript:document.f.reset();void(0);">
Установили значение по умолчанию</A>
```

В данном примере если кликнуть по гипертекстовой ссылке, то в форме происходит восстановление значений полей по умолчанию.

События объекта Form
Событие Submit

Событие *Submit* возникает (и соответствующий обработчик события onSubmit вызывается) при нажатии пользователем на кнопку типа*submit* или при выполнении метода *submit*(). Действие по умолчанию, которое выполняет браузер при возникновении этого события - отправка введенных в поля формы данных на сервер, указанный в атрибуте *ACTION*, с помощью метода, указанного в атрибуте *METHOD*, с использованием способа кодирования, указанного в атрибуте ENCTYPE, и с указанием того, что результаты работы *CGI-*

скрипта должны быть показаны в окне или фрейме с именем, указанным в атрибуте *TARGET*.

Фукцию обработки этого события можно переопределить и даже вовсе отменить. Для этой цели введен атрибутonSubmit="код_программы" у контейнера <FORM>. В нем можно указать действия (JavaScript-код), какие должны выполняться при возникновении этого события. Порядок выполнения этих действий и действий браузера, а также использование оператора return false для отмены последних, полностью аналогичны тем, что описаны ниже для onReset. Пример:

```
<SCRIPT>
function TestBeforeSend()
{
 if(document.f.query.value==")
 {
   alert('Пустую строку не принимаем!');
   return false;
 }
 else return true;
}
</SCRIPT>

<FORM NAME=f METHOD=post onSubmit="return TestBeforeSend();"
     ACTION="http://www.intuit.ru/rating_students/">
Ваше имя пользователя на intuit:<INPUT NAME=query>
<INPUT TYPE=submit VALUE="Посмотреть рейтинг">
</FORM>
```

В этом примере следует обратить внимание на конструкцию return TestBeforeSend(). Сама функция TestBeforeSend() возвращает значения true или false. Соответственно, данные либо отправляются на сервер, либо нет.

Событие Reset

Событие *Reset* возникает (и соответствующий обработчик события onReset вызывается) при нажатии пользователем на кнопку типа*reset* или при выполнении метода *reset*(). Действие по умолчанию, которое выполняет браузер при возникновении этого события - восстановление значений по умолчанию в полях формы. Однако функцию обработки этого события можно переопределить и даже вовсе отменить. Для этой цели введен атрибут onReset="код_программы" у контейнера <FORM>. В нем можно указать действия (JavaScript-код), какие должны выполняться при возникновении этого события. Браузер сначала выполняет эти действия, а затем - свое действие по умолчанию. Но если последним оператором в обработчике onReset будет return false, то действие браузера по умолчанию выполняться не будет. Этот прием называется *перехватом события*. Пример:

```
<FORM onReset="javascript: alert('Не дадим восстановить!');return false;">
<INPUT VALUE="Измените этот текст" SIZE=30>
<INPUT TYPE=reset VALUE="Восстановить">
</FORM>
```

Здесь команда return false предотвратила восстановление значения поля. Команда return true, равно как и отсутствие операторareturn, позволило бы браузеру продолжить *обработку события* - и восстановить значение поля.

Поля формы и их объекты

Как было сказано ранее, контейнеру <FORM> соответствует *объект* (назовем его f) класса Form ; он является свойством объектаdocument. В свою *очередь*, элементы формы, вложенные в *контейнер* <FORM>, например, <INPUT> различных типов, тоже соответствуют объектам различных классов, причем эти объекты являются *свойствами* объекта f.

У всех объектов, отвечающих полям формы, есть несколько стандартных свойств, доступных только для чтения: name (имя элемента, заданное в атрибуте NAME), type (тип элемента, например, для контейнеров <INPUT TYPE="..."> он совпадает со значением атрибута TYPE), form (указывает на форму f, в которой данный элемент содержится).

При программировании форм часто требуется писать обработчики событий для форм или их элементов, при этом нужно ссылаться на свойства данного элемента, других элементов и формы в целом. Стандартная схема именования по именам либо по индексам обсуждалась выше:

```
document.форма.элемент.свойство      // точечная нотация
document.форма.элемент["свойство"]   // скобочная нотация
document.forms["имя_формы"].elements["имя_элемента"].свойство
document.forms[индекс_формы].elements[индекс_элемента].свойство
```

Однако получающиеся выражения - довольно громоздкие. Поэтому было введено следующее **соглашение**: в обработчике события формы или элемента формы имя **текущего** элемента можно опускать (вместе со всей предшествующей "приставкой"). Кроме того, ссылаться на сам текущий элемент можно с помощью ключевого слова this. Последнее может потребоваться, например, когда нужно передать в функцию не какое-то *свойство* объекта, а сам *объект*.

Например, предположим, что у нас есть форма:

```
<FORM NAME=f>
<INPUT TYPE=text   NAME=e value="Текст"  onFocus="">
<INPUT TYPE=button NAME=b value="Кнопка" onClick="">
</FORM>
```

Тогда вместо полной записи:

```
<INPUT TYPE=text NAME=e value="Текст" onFocus="alert(document.f.e.value)">
```

мы можем использовать краткую, опустив приставку " document.f.e ", указывающую на текущий элемент:

```
<INPUT TYPE=text NAME=e value="Текст" onFocus="alert(value)">
```

Более того, в этом контексте эквивалентны следующие записи:

```
value  // короче не бывает!
          this.value  // здесь this ссылается на элемент "e"
        form.e.value  // form есть свойство объекта "e" (равное "f")
     this.form.e.value  // комбинируем оба способа
    document.f.e.value  // почти полная запись
window.document.f.e.value  // это самая полная запись
document.f.e.form.e.value  // можно итерировать "form.e."
```

Например, здесь в 3-й строчке form есть свойство (текущего!) элемента document.f.e - напомним, что это свойство ссылается на*объект* document.f. Аналогично, в обработчик *onClick* элемента b мы можем поместить *скрипт* form.e.value=50 (краткое обращение к свойству другого элемента формы: document.f.e.value) или *alert*(form.*method*) (краткое обращение к свойству самой формы document.f.*method*) или даже TestForBugs(this) (в *пользовательскую функцию* TestForBugs() будет передан (по ссылке) *объект* document.f.b).

Как видим, это соглашение не только дает некоторую экономию кода, но также позволяет ссылаться на текущий элемент или на форму, не зная его имени или номера. Это предоставляет дополнительную гибкость при программировании форм; например, можно переименовать форму, ничего не меняя во всех скриптах. Далее мы рассмотрим объекты JavaScript, соответствующие полям различных типов в *HTML-формах*. При этом мы будем пользоваться данным соглашением. Поскольку свойства name, type и form есть у объектов всех элементов формы, то мы не будем их указывать особо.

Текстовое поле ввода (объект Text)

Поля ввода (*контейнер* INPUT типа TYPE=text) являются одним из наиболее популярных объектов программирования на JavaScript. Это объясняется тем, что, помимо использования по прямому назначению, их применяют и в целях отладки программ, выводя в эти поля промежуточные значения переменных и свойств объектов.

```
<A HREF="http://site.com/">ссылка 1</A>
<FORM>Число гипертекстовых ссылок к данному моменту:
<SCRIPT>
document.write('<INPUT NAME=t VALUE='+document.links.length+'>');
</SCRIPT>
<BR><INPUT TYPE=button
VALUE="Число ссылок по окончании загрузки страницы"
onClick="form.t.value=document.links.length;">
<BR><INPUT TYPE=reset>
</FORM>
<A HREF="http://rite.com/">ссылка 2</A>
```

5.2.

В данном примере первое *поле* формы - это *поле ввода*. Мы присваиваем ему *значение* по умолчанию, равное числу гипертекстовых ссылок, имеющихся выше этого места в HTML-

документе. Затем при помощи кнопки изменяем это *значение* на общее количество гипертекстовых ссылок во всем HTML-документе.

С каждым текстовым полем ввода <INPUT TYPE=text> связан свой *объект* класса Text, который является свойством той формы, в которой он был описан. Этот *объект*, в свою *очередь*, характеризуется следующими свойствами, методами и событиями:

Свойства, методы и события объекта Text

Свойства	**Методы**	**Обработчики событий**	
defaultValue	focus()	onChange	onMouseOver
value	blur()	onSelect	onMouseOut
size	select()		onMouseDown
maxLength		onFocus	onMouseUp
		onBlur	
disabled			onKeyPress
readOnly		onClick	onKeyDown
		onDblClick	onKeyUp

Все перечисленные *свойства* можно менять. Смысл их таков: value (текущее *значение поля ввода*), defaultValue (*значение поля ввода* по умолчанию), size (число умещающихся в *поле* символов, т.е. видимых) maxLength (максимальное число символов, которое можно присвоить значению данного поля) *readOnly* (может ли *пользователь* менять *значение* поля) *disabled* (может ли*пользователь* установить фокус на этом *поле*).

Опишем вкратце *методы*: *focus*() - устанавливает фокус на данном *поле*, *blur*() - убирает фокус с данного поля, select() - выделяет весь введенный текст (чтобы, например, его можно было скопировать в *буфер*, либо удалить, нажав клавишу *Delete*).

Смысл *обработчиков событий* вполне понятен из их названий: обработчик *onChange* вызывается, когда *пользователь* (но не *скрипт*) изменил *значение* в *поле ввода* (и кликнул вне *поля ввода*); onSelect - когда *пользователь* начинает выделять текст, расположенный в*поле*; onFocus и onBlur - когда *поле* получает и теряет фокус, соответственно; *onClick* и *onDblClick* - когда *пользователь*совершил одинарный или *двойной щелчок* мышью на *поле*, соответственно. Вторая колонка событий - стандартна для большинства элементов HTML-страницы. Нужно лишь иметь в виду, что обработчики событий onMouseDown, onMouseUp, onKeyPress, onKeyDown,onKeyUp срабатывают у того элемента формы, который в данный момент находится в фокусе.

Кнопки

В *HTML-формах* используется четыре вида кнопок:

```
<FORM>
<INPUT TYPE=button VALUE="Кнопка типа button">
<INPUT TYPE=submit VALUE="Кнопка отправки">
<INPUT TYPE=reset  VALUE="Кнопка сброса">
```

```
<INPUT TYPE=image SRC=a.gif> <!-- графическая кнопка -->
</FORM>
```

В атрибуте кнопки можно задать обработчик события *onClick*, а в атрибуте формы - обработчики событий onSubmit и onReset. Кроме того, кнопкам и форме соответствуют объекты *DOM*. *Объект*, отвечающий кнопке, имеет метод *click*(). *Объект*, отвечающий форме, имеет методы *submit*() и *reset*(). С точки зрения программирования важен вопрос о взаимодействии этих методов друг с другом и с соответствующими обработчиками событий.

В каком случае при вызове метода (из любого места JavaScript-программы) будет автоматически вызван и соответствующий обработчик события, заданный пользователем в атрибуте кнопки или формы? Ответ здесь следующий:

- при вызове метода *click*() кнопки вызывается и обработчик события *onClick* этой формы;
- при вызове метода *submit*() формы **не вызывается** обработчик события onSubmit формы;
- при вызове метода *reset*() формы вызывается и обработчик события onReset формы.

Ниже мы на примерах рассмотрим, что это означает на практике. Таким образом, при программном вызове метода *submit*() нужно позаботиться о дополнительном вызове обработчика события onSubmit, чтобы, например, данные не были отправлены на *сервер* без предварительной проверки. Как это сделать - мы расскажем ниже. Особое внимание мы уделим также возможности перехвата и генерирования события *отправки данных* на *сервер*.

Кнопка button

Кнопка типа button вводится в форму главным образом для того, чтобы можно было выполнить какие-либо действия либо при ее нажатии пользователем, либо при вызове метода *click*().

```
<FORM NAME=f>
<INPUT TYPE=button NAME=b VALUE="Кнопка" onClick="alert('5+7='+(5+7))">
</FORM>
<A HREF="javascript:document.f.b.click();void(0);">Вызвать метод click()</A>
```

Вызов метода click() у кнопки равносилен нажатию кнопки, что и демонстрирует приведенный пример. Как мы увидим ниже, это же справедливо для любых типов кнопок.

Кнопка submit

Кнопка отправки (*submit*) позволяет отправить данные, введенные в форму, на сервер. В простейшем случае - при отсутствии у контейнера <FORM> атрибутов *ACTION* (его значением по умолчанию является адрес текущей страницы), *METHOD* (его значением по умолчанию является GET) и *TARGET* (его значением по умолчанию является _self) - стандартным действием браузера при *отправке данных* на сервер является просто перезагрузка текущей страницы, что подтверждает следующий пример:

```
<FORM>
```

```
<INPUT TYPE=submit>
</FORM>
```

Для имитации ответа сервера заготовим следующий простой HTML-файл *receive.htm*:

```
<HTML><BODY>Данные приняты!</BODY></HTML>
```

Теперь усложним наш пример: добавим обработчики событий *onClick* (у кнопки отправки) и onSubmit (у формы), и посмотрим на поведение браузера при нажатии кнопки отправки:

```
<FORM NAME=f ACTION="receive.htm"
    onSubmit="return confirm('Вы хотите отправить данные?')">
<INPUT onClick="alert('Вызван обработчик onClick у кнопки отправки')"
     TYPE=submit VALUE="Кнопка отправки" NAME=s>
</FORM>
```

5.6. Обработчики onClick у кнопки отправки и onSubmit у формы

Убедитесь, что нажатие кнопки отправки приводит к следующей последовательности действий браузера:

1. вызов обработчика события *onClick* у данной кнопки;
2. вызов обработчика события onSubmit у формы;
3. *отправка данных* формы на сервер.

Соответственно, для выполнения дополнительных действий перед *отправкой данных* можно поместить код в любой из указанных обработчиков; в частности, поместив в какой-либо из них оператор return false, мы сможем предотвратить *отправку данных*.

Вызов метода click() кнопки отправки равносилен нажатию этой кнопки - произойдут все три вышеперечисленных действия:

```
<FORM NAME=f ACTION="receive.htm"
    onSubmit="return confirm('Вы хотите отправить данные?')">
<INPUT onClick="alert('Вызван обработчик onClick у кнопки отправки')"
     TYPE=submit VALUE="Кнопка отправки" NAME=s></FORM>

<A HREF="javascript: document.f.s.click();void(0);"
 >Вызвать метод <B>click()</B> кнопки отправки</A>
```

5.7. Вызов метода click() у кнопки отправки

Метод submit() формы

Вызов метода submit() формы **не равносилен** нажатию кнопки отправки. При вызове этого метода будет выполнено только третье из вышеперечисленных трех действий - *отправка данных* на сервер. То, что он не должен порождать вызов обработчика *onClick* кнопки отправки, вполне понятно - ведь мы пытаемся отправить данные в обход кнопки отправки (которой, кстати, может и не быть вовсе). Но и обработчик события onSubmit у формы тоже **не вызывается** - это является для многих неожиданным. Не будем судить, насколько это логично (и почему это поведение отличается от поведения метода *reset*(), см. ниже), а

просто проиллюстрируем этот эффект, введя в предыдущий пример ссылку, вызывающую метод *submit*():

```
<FORM NAME=f ACTION="receive.htm"
     onSubmit="return confirm('Вы хотите отправить данные?')">
<INPUT onClick="alert('Вызван обработчик onClick у кнопки отправки')"
      TYPE=submit VALUE="Кнопка отправки" NAME=s></FORM>

<A HREF="javascript: document.f.submit();void(0);"
 >Вызвать метод <B>submit()</B> формы</A>
```

5.8. Метод submit() не вызывает обработчика onSubmit

Тем самым данные могут уйти на сервер без предварительной проверки JavaScript-скриптом. Каким же образом заставить браузер вызвать обработчик onSubmit? Для этого существует возможность обратиться к этому обработчику напрямую: document.f.onsubmit(). Остается предусмотреть, что после этого метод *submit*() должен вызываться не всегда, а только если onSubmit либо не возвратил никакого значения, либо возвратил true, иными словами, если он не возвратил false. Окончательно мы получаем:

```
<FORM NAME=f ACTION="receive.htm"
     onSubmit="return confirm('Вы хотите отправить данные?')">
<INPUT onClick="alert('Вызван обработчик onClick у кнопки отправки')"
      TYPE=submit VALUE="Кнопка отправки" NAME=s></FORM>

<A HREF="javascript:
   if(document.f.onsubmit() != false)
     document.f.submit();  void(0);"
 >Вызвать <B>submit()</B> с преварительной проверкой onSubmit</A>
```

5.9. Принудительный вызов onSubmit перед submit()

Есть еще один способ инициировать *отправку данных* формы в обход кнопки отправки (которой, кстати, у формы может и не быть). Если фокус находится на любом текстовом поле <INPUT TYPE=text> формы и пользователь нажмет клавишу Enter, то (в большинстве браузеров) произойдет вызов обработчика события onSubmit формы и *отправка данных* на сервер.

```
Введите текст и нажмите Enter:<BR>
<FORM ACTION="receive.htm"
  onSubmit="return confirm('Вы хотите отправить данные?')">
<INPUT TYPE=text VALUE="Текст вводить здесь:" SIZE=50>
</FORM>
```

5.10. Отправка данных формы нажатием клавиши Enter

Этот способ работает логичнее, чем метод *submit*(), т.к. отправляемые на сервер данные не избегают предварительной проверки обработчиком onSubmit.

Кнопка reset

Кнопка сброса (*reset*) позволяет вернуть все поля формы в первоначальное состояние, которое они имели при загрузке страницы. Нажатие кнопки сброса приводит к следующей последовательности действий браузера:

1. вызов обработчика события *onClick* у данной кнопки;
2. вызов обработчика события onReset у формы;
3. восстановление значений по умолчанию во всех полях формы.

Вызов метода click() у кнопки сброса равносилен нажатию этой кнопки, т.е. приводит к тем же трем действиям:

```
<FORM NAME=f
  onReset="return confirm('Вы хотите очистить форму?')">
<INPUT TYPE=text VALUE="Измените этот текст">
<INPUT TYPE=reset VALUE="Кнопка сброса" NAME=s
 onClick="alert('Вызван обработчик onClick у кнопки сброса')">
</FORM>
<A HREF="javascript: document.f.s.click();void(0);"
 >Вызвать метод <B>click()</B> кнопки сброса</A>
```

5.11. Вызов метода click() у кнопки сброса

Есть способы сбросить форму в исходное состояние в обход кнопки сброса (которой, кстати, у формы может и не быть). Во-первых, это*вызов метода reset*() у формы. Во-вторых, если фокус находится на любом поле или кнопке формы, то можно нажать клавишу Esc. Пример:

```
Измените текст, а затем нажмите Esc (либо ссылку).<BR>
<FORM NAME=f
  onReset="return confirm('Вы хотите очистить форму?')">
<INPUT TYPE=text VALUE="Измените этот текст">
</FORM>
<A HREF="javascript: document.f.reset();void(0);"
 >Вызвать метод <B>reset()</B> формы</A>
```

5.12. Сброс формы нажатием клавиши Esc

Как можно видеть, оба способа не просто сбрасывают форму, но и вызывают обработчик события onReset формы. Таким образом, метод *reset*() ведет себя более логично и *предсказуемо*, нежели *submit*().

Графическая кнопка

Графическая кнопка - это разновидность кнопки отправки. Ее отличие в том, что вместо кнопки с надписью пользователь увидит картинку, по которой можно кликнуть:

```
<FORM ACTION="receive.htm">
<INPUT TYPE=image SRC="pic.gif">
</FORM>
```

Кроме того, когда пользователь *кликает* по графической кнопке, то на сервер отправятся не только данные, введенные в поля формы, но также и координаты указателя мыши относительно левого верхнего угла изображения. К сожалению, перехватить эти координаты в JavaScript-программе не удается. Если Вам необходимо работать с этими координатами, то вместо графической кнопки рекомендуется создать активную карту с помощью контейнера <MAP>.

Графические кнопки имеют ряд странностей. Например, являясь одновременно и кнопкой, и изображением, они почему-то отсутствуют как в коллекции document.f.*element*s[], так и в коллекции document.images[] (IE 7, Mozilla Firefox). Как следствие, они не учитываются ни в общем количестве элементов формы (document.f.*length*), ни в общем количестве изображений документа (document.images.*length*).

Как же обратиться к такой кнопке? Это можно сделать, например, задав атрибут ID:

<INPUT TYPE=image SRC="pic.gif" ID="d1">

и затем в программе написав: var knopka = document.*getElementById*('d1'). После этого мы можем обращаться к свойствам этой кнопки, например knopka.src, а также к методу knopka.*click*(). Следующий пример показывает, что *вызов метода click*()графической кнопки "почти" равносилен нажатию этой кнопки, т.е. последовательно вызывает обработчики *onClick* кнопки,onSubmit формы и передает данные на сервер (но что при этом передается в качестве координат курсора мыши?):

```
<FORM ACTION="receive.htm"
    onSubmit="return confirm('Вы хотите отправить данные?')">
<INPUT onClick="alert('Вызван обработчик onClick у графической кнопки')"
    TYPE="image" SRC="pic.gif" id="d1">
</FORM>

<A HREF="javascript:
  var knopka = document.getElementById('d1');
     knopka.click();  void(0);"
 >Вызвать метод <B>click()</B> графической кнопки</A>
```

5.13. Вызов метода click() у графической кнопки

Поскольку графические кнопки используются довольно редко, на этом мы остановимся в их обсуждении.

Практическая работа №34
Работа с графическими изображениями
Цель: освоить средства работы с графическими изображениями для веб.

Пакет GIF Construction Set для Windows является мощным средством для работы с GIF-файлами, содержащими несколько блоков. Пакет позволяет:

- создавать файлы, содержащие несколько изображений, из существующих отдельных изображений;

- добавлять, редактировать и удалять блоки комментариев, блоки с текстом, накладывающимся на изображение;
- сохранять изображения в режиме чередования строк или построчном режиме;
- задавать признак прозрачности цвета для отдельных изображений;
- задавать параметры временных задержек, а также цикла анимации для Netscape, которые также распознаются Microsoft Internet Explorer;
- просматривать отдельные изображения;
- импортировать файлы, хранящие изображения во многих популярных форматах, в том числе и 24-битных;
- изменять параметры месторасположения отдельных изображений на логическом экране.

Этот программный продукт существует уже очень давно и, как многие программы, многократно видоизменялся. На настоящий момент последней является версия 2.0a, датированная августом 1999 года. Эта версия сохранила все функциональные возможности предыдущих версий, однако имеет несколько отличный интерфейс. Для большинства задач вполне достаточно наличия и более ранних версий. Так, примеры будут даваться для версий от 1.0K до 1.0P, применение которых вполне допустимо и сегодня, несмотря на то, что появление этих версий относится к 1996 году.

Отличительной особенностью программы является простота работы с ней. Освоение типовых действий для создания анимации требует весьма небольших затрат времени.

Рассмотрим основные возможности программы и методы работы с ней. После запуска программы GIF Construction Set и считывания GIF-файла на экран выдается список блоков файла (рис. 3.15).

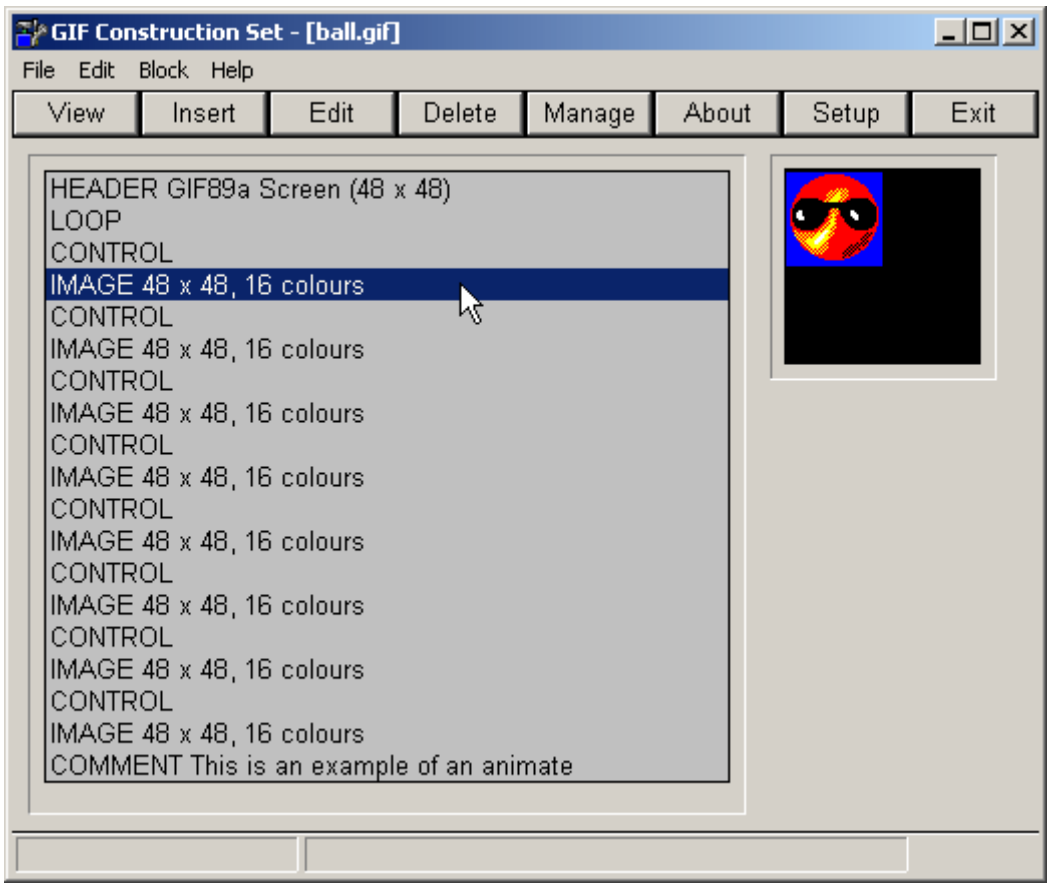


Рис. *Список блоков GIF-файла*

Файл всегда содержит один блок заголовка (HEADER) и один или несколько блоков с отдельными изображениями IMAGE. Остальные блоки могут отсутствовать. В правом верхнем углу окна программы отводится место для быстрого просмотра изображений в уменьшенном виде. Предоставляется возможность удалить, добавить или отредактировать параметры любого из блоков.

Блок HEADER

Блок HEADER (рис. 3.16) содержит информацию о размерах логического экрана в пикселах. Эти параметры не влияют на размеры самих изображений, но для Web-браузеров определяют размеры прямоугольной области, внутри которой будут располагаться все изображения данного GIF-файла. Каждое изображение имеет определенный размер по горизонтали и вертикали и заданное смещение в пикселах от левого верхнего угла логического экрана. В простейшем случае смещение равно нулю, и тогда размер логического экрана должен быть равен максимальному из размеров всех изображений отдельно по горизонтали и по вертикали. При смещении, большем нуля, к размеру изображения добавляется величина смещения. Пользователь может самостоятельно задать размеры логического экрана, однако следует иметь в виду, что отображение картинок, выходящих за пределы логического экрана в Netscape Navigator, вызывает неустранимую ошибку GPF (General Protection Fault). Поэтому программа GIF Construction Set при сохранении файла автоматически вычисляет необходимые размеры логического экрана с учетом размеров отдельных изображений и их смещений независимо от значений, заданных пользователем.

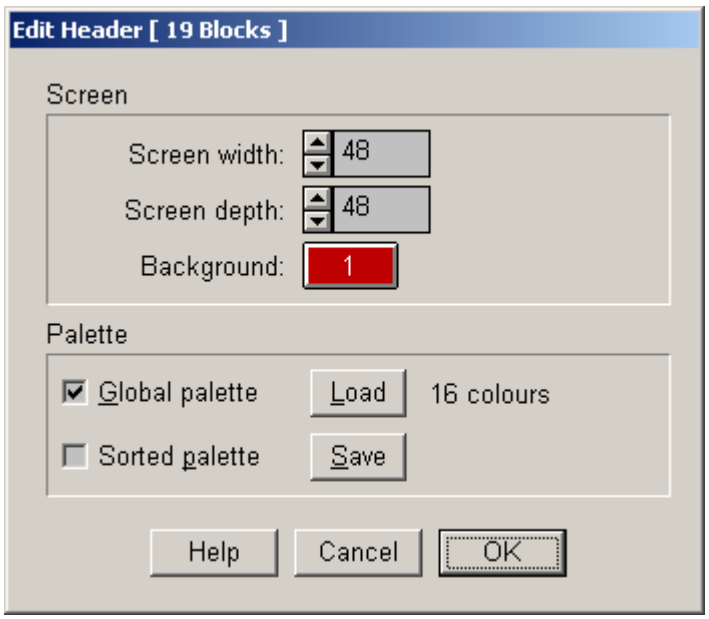


Рис. *Задание общих параметров GIF-файла*

Блок LOOP

Блок LOOP (рис. 3.17) определяет параметры цикла анимации для Netscape. Блок содержит единственный параметр — число повторений цикла, которое можно изменять в пределах от нуля до 32760. Нулевое значение определяет бесконечный цикл. Выдаваемое сообщение о том, что Netscape игнорирует значение счетчика цикла, уже неактуально для Netscape 3.0 и выше. Блок LOOP может отсутствовать, при этом все изображения будут выданы один раз, и на экране останется последнее. При наличии блока LOOP он всегда располагается вторым в списке блоков после HEADER.

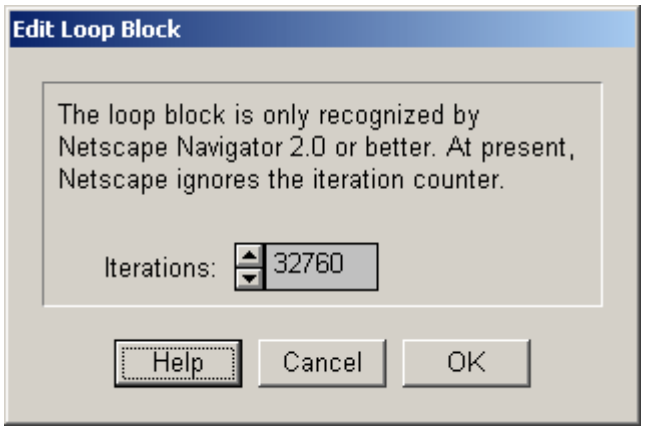


Рис. 3.17. *Задание параметров цикла анимации*

Блок CONTROL

Перед каждым блоком с описанием изображения IMAGE может располагаться блок CONTROL с управляющими параметрами (рис. 3.18) для последующего блока IMAGE. Задается флаг наличия прозрачного цвета (Transparent colour) и номер этого цвета, который

вводится вручную или указывается непосредственно на рисунке при помощи специального маркера, пиктограмма которого представлена в правом верхнем углу. При указании маркером прозрачным становится цвет того пиксела, который был отмечен на изображении. Так как обычно в качестве прозрачного выбирается цвет фона рисунка, занимающего значительную площадь изображения, то проблема попадания в пиксел нужного цвета не возникает.

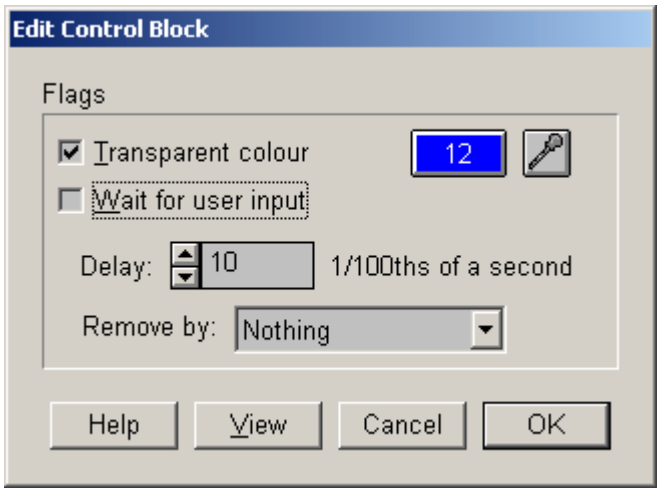


Рис. 3.18. *Задание параметров блока управления*

Следующий параметр блока (Wait for user input) задает ожидание нажатия клавиши пользователем перед сменой изображения. Браузеры не поддерживают это свойство, и поэтому значение данного параметра не играет роли.

Параметр Delay задает время задержки данного изображения в сотых долях секунды. Заметим, что реальное время задержки при отображении в браузере может быть больше указанного, например, при нескольких файлах GIF со сменяющимися изображениями на экране или при недостаточной скорости процессора и/или возможностей видеокарты. При разработке HTML-документов следует учитывать, что выполнения операций по смене изображений могут потребоваться значительные затраты мощностей процессора, что замедлит выполнение других операций и создаст неудобства в работе. Рекомендуется вставлять задержки в несколько секунд в конец цикла анимации для освобождения процессора для выполнения других операций.

Приведем результаты эксперимента по анализу загрузки процессора при выполнении операций такого рода. Был подготовлен GIF-файл, содержащий 5 изображений размером 150x174. Для смены изображений установлен бесконечный цикл. Исследовалась загрузка процессора при отображении этого файла браузером в отсутствии других задач. Результаты исследования определяются большим числом факторов, поэтому приведем все необходимые сведения об условиях проведения эксперимента. Отображение выполнялось браузером Netscape 4.7 в режиме 800x600x256. Процессор Pentium-200 MMX, RAM 64 Мб, видеокарта S3 Virge с 2 Мб памяти. Операционная система Windows 95. Проанализировано два варианта отображения GIF-файла, которые отличаются величинами временных задержек между сменой изображений. Загрузка процессора изучалась с помощью программы "Системный монитор", входящей в состав операционной системы. Результаты анализа показаны на рис. 3.19. Представлена загрузка процессора в процентах при

временных отсчетах, равных 5 с. Левая часть графика (примерно одна четверть всего временного интервала) характеризует загрузку процессора при отображении файла с изображениями, сменяющимися без задержек. Видно, что загрузка составляет величину порядка 40%. Скорее всего, при наличии более быстродействующей видеокарты загрузка процессора была бы еще больше. Далее график показывает загрузку процессора при отображении файла с изображениями, при смене которых задана задержка, равная 0,1 с, а в конце цикла смены изображений установлена задержка 2 с. Загрузка процессора составила примерно 10%. В самом конце графика загрузка упала практически до нуля (осталось 2%), что произошло при нажатии кнопки Stop в браузере и прекращении цикла анимации.

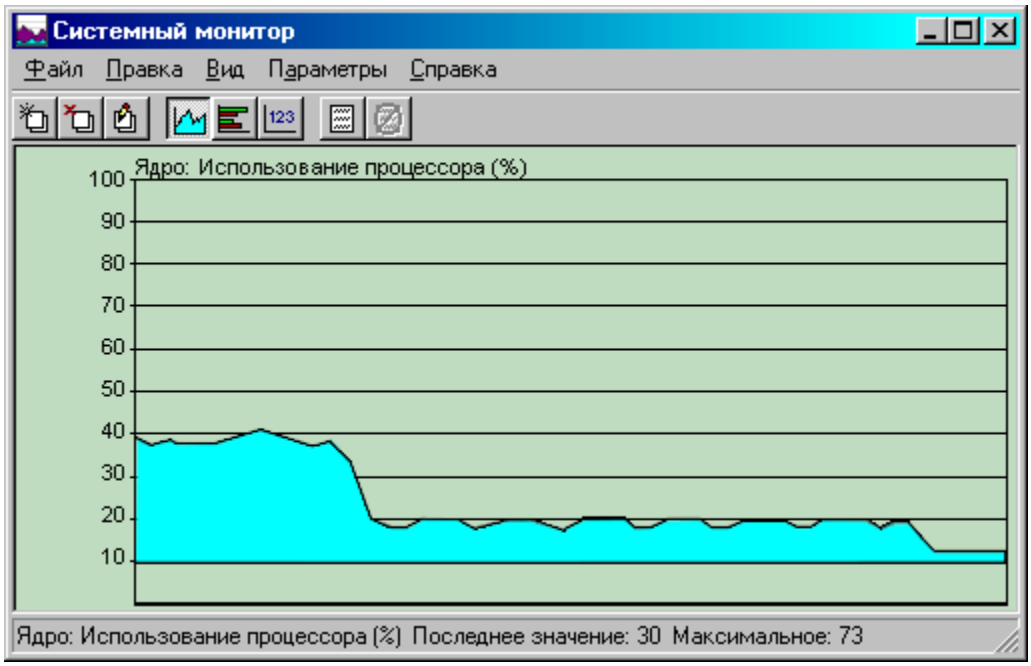


Рис. 3.19. *Загрузка процессора при отображении циклически сменяющихся изображений с различными задержками*

Из этого примера можно сделать вывод, что неразумное задание параметров анимации может привести к значительной трате ресурсов процессора, часто бесполезной. Несколько анимированных GIF-файлов на странице с минимальными задержками, и неудобства при просмотре обеспечены. К сожалению, в браузерах отсутствует возможность настройки режима демонстрации анимированных изображений. Можно либо не загружать изображения вообще, либо вручную останавливать загрузку кнопкой Stop, что приведет, в том числе, и к остановке циклов анимации.

Примечание

Для браузера Netscape известны хакерские приемы, останавливающие анимацию после прокрутки одного цикла. Для некоторых конкретных версий браузера известно, в каких адресах EXE-файла браузера нужно внести изменения, в результате которых браузер будет выполнять цикл анимации только один раз независимо от настроек в файле с изображениями. Конечно, подобные приемы нельзя пропагандировать, но раз это делается, значит, актуальность налицо.

Параметр Remove by (рис. 3.20) определяет способ восстановления данных на месте изображения после завершения времени задержки изображения. Напомним, что режим Previous image (восстановление предыдущего состояния фрагмента экрана) не реализуется браузерами. Типичным вариантом восстановления является режим Background, обеспечивающий заполнение цветом фона или фоновым изображением, которое в сочетании с прозрачным цветом и изменяющимся смещением внутри логического экрана для отдельных кадров может создать хорошую иллюзию перемещающегося изображения. Если все изображения GIF-файла одинакового размера, имеют одинаковое смещение и не имеют прозрачного цвета, то в качестве режима восстановления достаточно указать Nothing (ничего не делать) или Leave as is (оставить как есть).

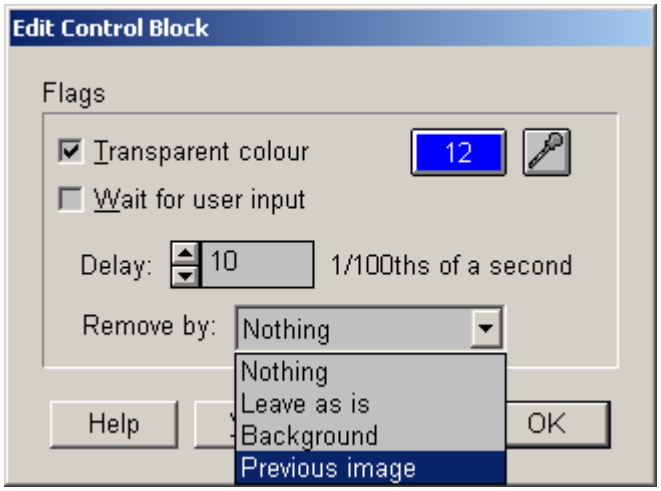


Рис. 3.20. *Задание способа восстановления изображения*

Примечание

Хотя для параметра Remove by можно задавать четыре различных значения, по существу предоставляется всего две возможности: режим Background или любой другой режим из трех оставшихся, которые реализуются браузерами одинаково.

Блок IMAGE

Блок IMAGE непосредственно содержит информацию об отдельном изображении (рис. 3.21). Размеры изображения (Image width и Image depth) приводятся в качестве справочных, их изменение здесь невозможно, а параметры Image left и Image top определяют смещение в пикселах данного изображения относительно левого верхнего угла логического экрана, которое задается пользователем. В данном блоке также задается тип хранения изображения — с чередованием строк (Interlaced) или без, наличие локальной палитры (Local Palette) и заголовок блока (Block title), который не отражается при выводе и является краткой текстовой характеристикой изображения. Использование локальной палитры для отдельного изображения не рекомендуется. Каждая локальная палитра увеличивает размер файла на 779 байт и может ухудшить отображение цвета при смене кадров. При сохранении файла пакет GIF Construction Set предупреждает о наличии локальной палитры в одном из блоков и предлагает выполнить ее приведение к глобальной.

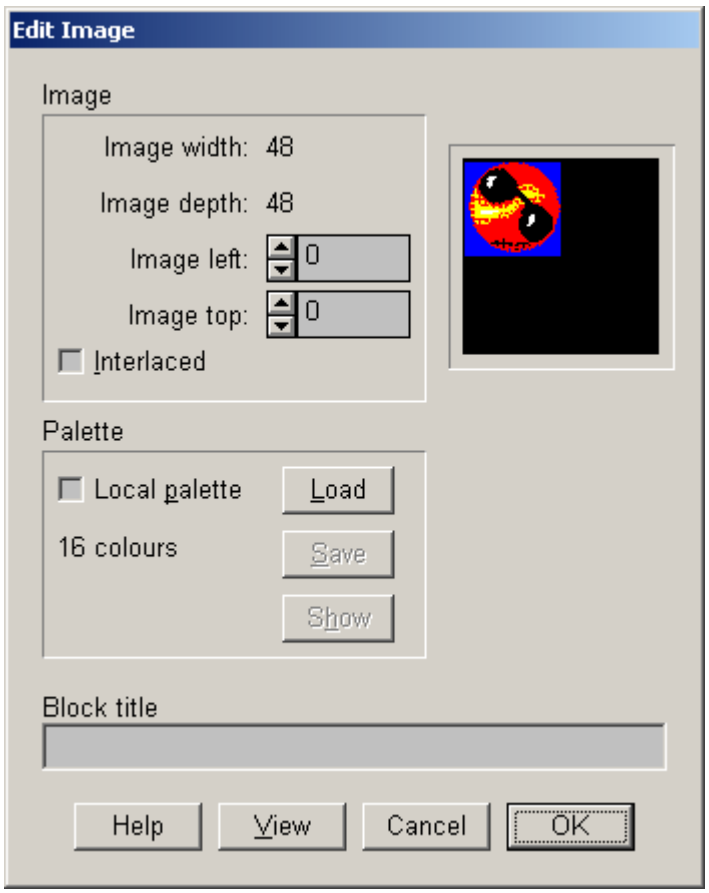


Рис. 3.21. *Окно редактирования параметров изображения*

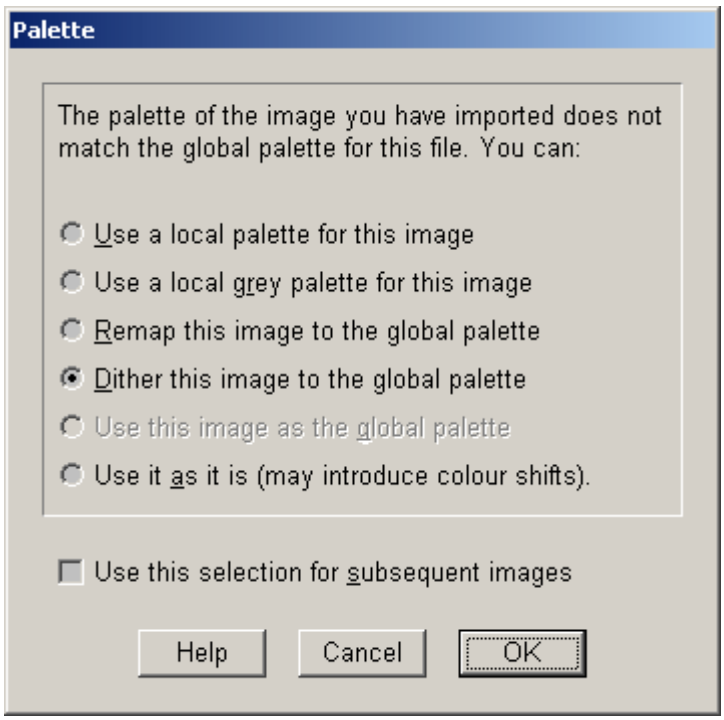


Рис. 3.22. *Окно выбора способа приведения палитры*

При добавлении (Insert) изображения в список блоков файла выполняется сравнение глобальной палитры с палитрой добавляемого изображения. Предварительно для 24-битовых изображений выполняется преобразование к 256 цветам. Если какие-либо цвета отсутствуют в глобальной палитре, то предлагается сделать выбор варианта преобразования (рис. 3.22). Не углубляясь в детали, отметим, что для фотореалистичных изображений наилучшим выбором является метод Dither, при котором делается попытка имитировать цвет, отсутствующий в палитре совокупностью нескольких разноцветных пикселов из палитры. Для иных изображений обычно достаточно метода Remap, при котором отсутствующие цвета просто заменяются на ближайшие из палитры.

Использование мастера анимации

Создание анимации из отдельных кадров может быть осуществлено вручную использованием пунктов меню пакета, описанных выше. Однако можно быстро создать нужную анимацию с использованием мастера анимации Animation Wizard, входящего в состав программы GIF Construction Set. Вся процедура создания заключается в ответе на ряд вопросов, последовательно задаваемых пользователю. Отдельные кадры этой процедуры показаны на рис. 3.23 и рис. 3.24.

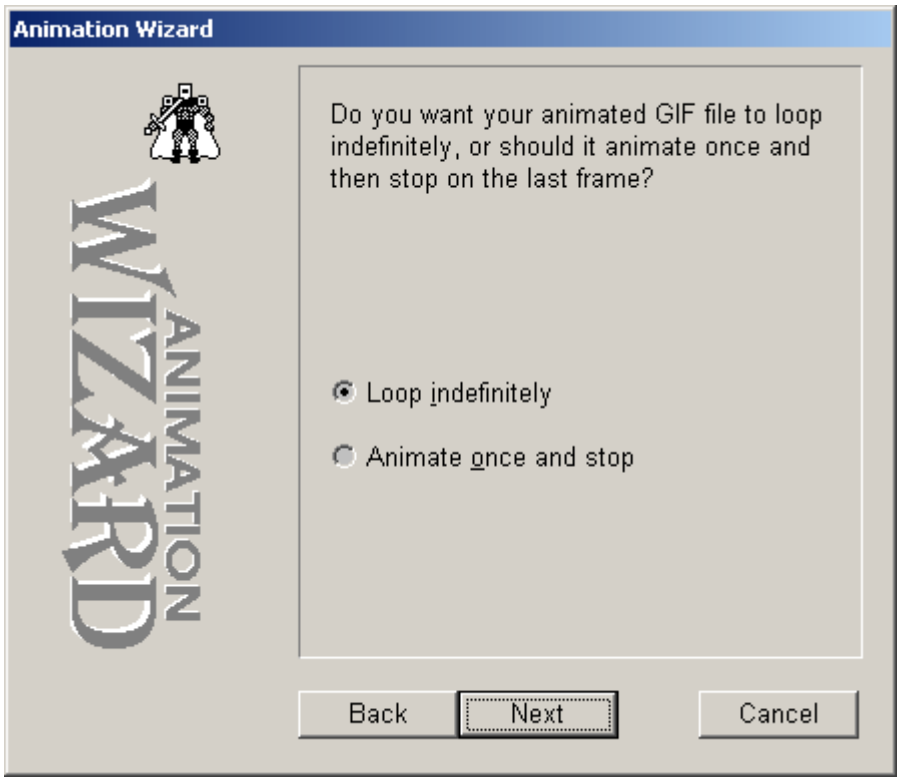


Рис. 3.23. *Подготовка анимации с помощью мастера Animation Wizard*

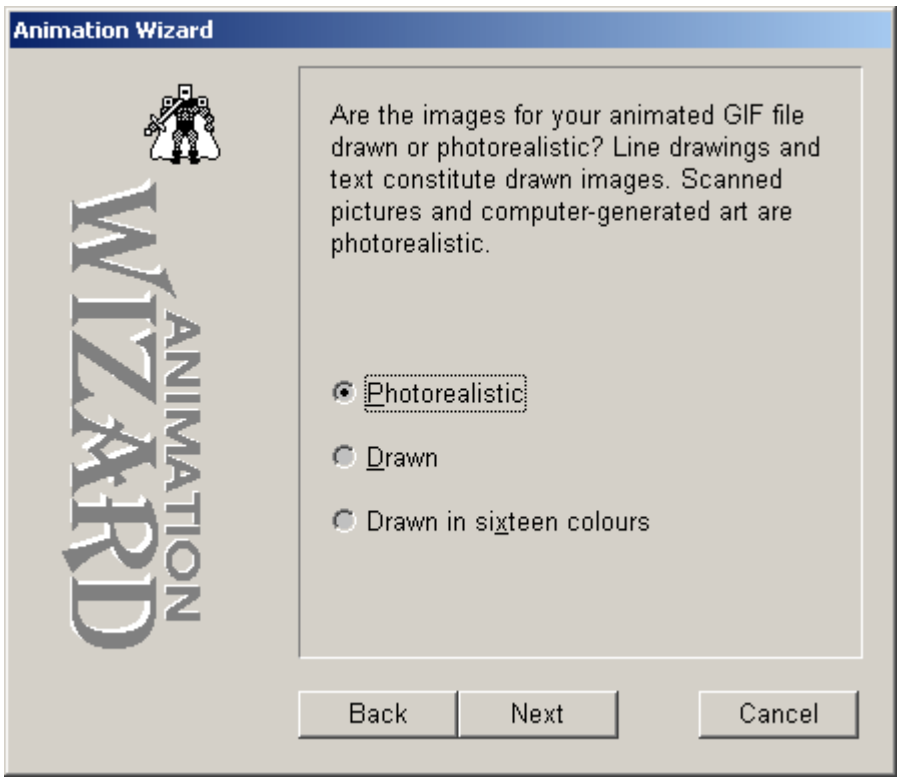


Рис. 3.24. *Один из шагов подготовки анимации*

Ускорить процесс задания параметров кадров анимации можно с использованием пункта меню Manage (рис. 3.25), который позволяет определить ряд общих значений параметров для нескольких кадров за одну операцию.

Еще одна возможность, предоставляемая пакетом, заключается в преобразовании анимации в формате AVI в GIF-файл с набором кадров. Преобразование может занять значительное время и создать файл очень большого размера. Для AVI-файлов типичны размеры в единицы и десятки мегабайт, однако GIF-файлы такого размера, как правило, не используются.

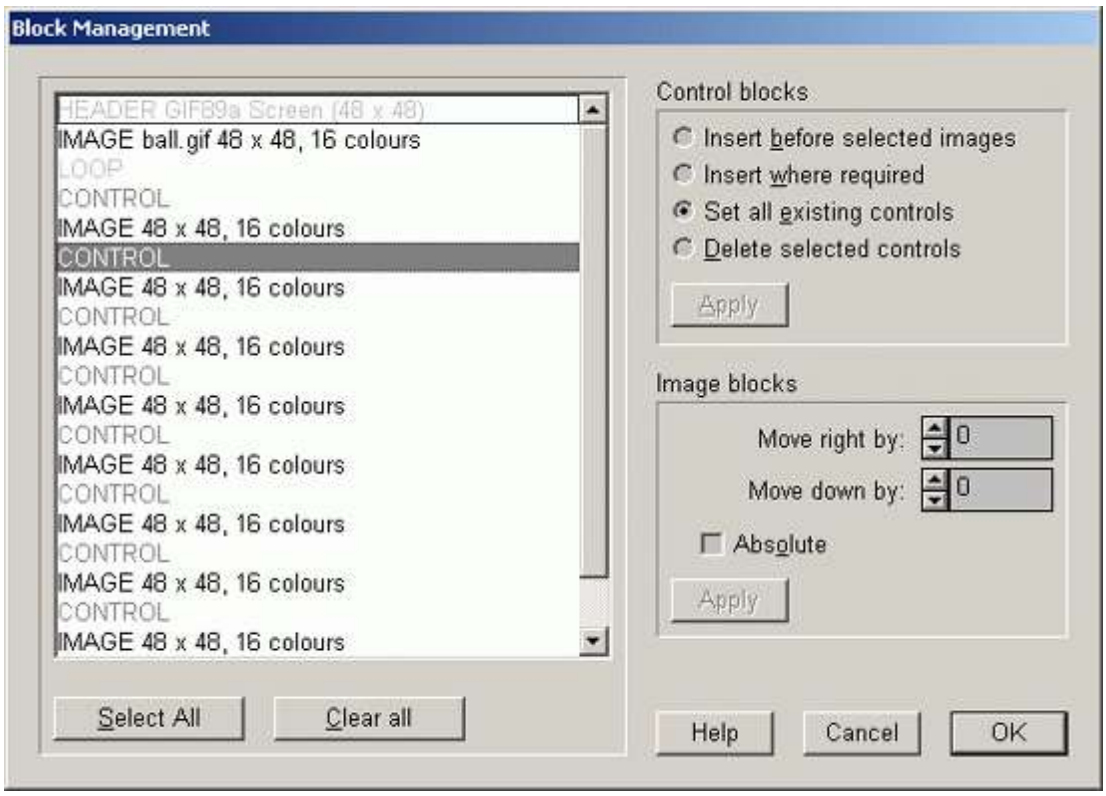


Рис. 3.25. *Установка общих свойств блоков*

В список блоков, составляющих файл GIF, можно добавлять и ряд других (рис. 3.26), но создание и редактирование произвольных блоков, например, специфичных для какого-либо приложения, не предусмотрено.

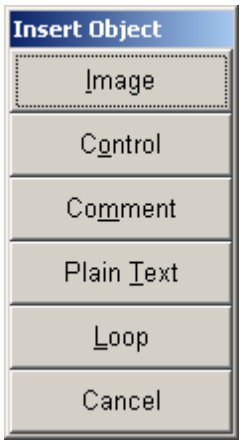


Рис. 3.26. *Окно вставки блоков*

Заметим, что сохранение файла программой GIF Construction Set всегда производится с дескриптором GIF89a.

Дополнительные возможности пакета GIF Construction Set

Современные версии пакета GIF Construction Set обладают некоторыми дополнительными возможностями, которые позволяют создавать GIF-файлы специального содержания. Эти возможности можно найти в пункте меню

Edit пакета. Заметим, что реализация описываемых ниже возможностей связана не с какими-либо конкретными особенностями формата файла GIF или браузеров, которые не были использованы до этого, а с некоторыми сервисными инструментами быстрого создания файлов с множественными изображениями. Перечислим эти возможности в порядке их появления в меню. Отметим, что в таком же порядке они появлялись с развитием пакета от версии 1.0K, в которой не было ни одной из них, и до версии 1.0P, начиная с которой в программу были включены все описываемые ниже пункты.

Banner. Создание бегущей строки. В действительности пакет генерирует ряд изображений, содержащий текст или его фрагменты, которые по мере смены отдельных изображений создают иллюзию бегущей строки.

Transition. Имитация одного из выбираемых методов появления изображения на экране. Различные методы появления изображений широко используются в ряде пакетов работы с графикой. Одним из примеров является пакет презентационной графики PowerPoint, входящий в состав Microsoft Office. В отличие от таких пакетов Web-браузеры не имеют средств для реализации методов появления изображения (не считая вариантов формата GIF — построчное хранение или чередование строк и вариантов формата JPG — обычный или прогрессивный). Использование данного пункта меню позволяет быстро создать ряд изображений, содержащих отдельные фазы смены изображения с заданными временными задержками и числом кадров.

Wide Palette GIF. Использование палитры, содержащей более чем 256 одновременно используемых цветов. Название этого пункта может привести в недоумение, поэтому сразу же еще раз подчеркнем, что структура файла GIF не позволяет хранить изображение, имеющее более 256 цветов. Однако внутри одного GIF-файла может размещаться несколько изображений, каждое из которых имеет свою локальную палитру. Это свойство применяется для искусственного увеличения числа используемых цветов. Все изображение разбивается на ряд отдельных кадров одинакового размера в зависимости от количества используемых цветов. Например, для 510 цветов достаточно двух кадров. Каждый из кадров имеет свою палитру и включает пикселы, цвет которых соответствует локальной палитре данного кадра. Остальные пикселы задаются одним определенным цветом, который объявляется прозрачным. Таким образом, каждый кадр может содержать до 255 уникальных цветов, один цвет отводится под прозрачный. При последовательном выводе нескольких кадров на одно и то же место изменяются лишь те пикселы, цвет которых отличен от прозрачного. Вывод всей последовательности кадров обеспечит требуемое разнообразие цветов. Такие изображения не могут иметь одного прозрачного цвета, поскольку прозрачность используется для специальных целей, как описано выше. Изображения с расширенной палитрой в принципе могут быть анимированными, однако это приведет к медленной смене кадров, так как каждый из них, по существу, состоит из нескольких. Исходное изображение должно содержать 24-битовый цвет, например, это может быть файл формата BMP, PCX или TIFF. Отметим, что данный режим поддерживается только 32-битной версией пакета. Эффект большого количества цветов хорошо смотрится только в режиме монитора HiColor или TrueColor, при просмотре в режиме 256 цветов изображение может оказаться просто ужасным. Все эти особенности

заставляют подходить к использованию режима расширенной палитры с большой осторожностью.

LED Sign. Создание бегущей строки, напоминающей по характеру световую рекламу или табло электронных приборов. В тексте строки при помощи специальных знаков задается цвет символов. Длина строки с учетом спецзнаков не может превосходить 260 символов.

В целом все описанные возможности приводят к созданию файлов с множественными изображениями, что может значительно увеличить их размер. Получаемый при этом эффект следует сопоставлять с дополнительными затратами на его осуществление, состоящими в увеличении времени передачи

файлов по сети и общего трафика. Отдельные Web-страницы с излишне большим количеством эффектов часто свидетельствуют только об отсутствии чувства меры у создателей и лишь затрудняют восприятие.

Практическая работа №35
Основы синтаксиса PHP. Оформление кода программы. Переменные, константы и типы данных языка программирования PHP.

Цель: изучить основы синтаксиса PHP, оформление кода программы, переменные, константы и типы данных языка программирования PHP на конкретных примерах

Общие понятия
Язык PHP специально предназначен для веб-программирования. PHP сочетает достоинства языков C и Perl и при этом весьма прост в изучении и обладает значительными преимуществами перед традиционными языками программирования.
Синтаксис PHP очень напоминает синтаксис языка C и во многом заимствован из таких языков как Java и Perl.
Программист C очень быстро освоит язык PHP и сможет использовать его с максимальной эффективностью.
В принципе, в PHP есть практически все операторы и функции, имеющиеся в стандартном GNU C (или их аналоги), например есть циклы (while, for), операторы выбора (if, switch), функции работы с файловой системой и процессами (fopen, *dir, stat, unlink, popen, exec), функции ввода-вывода (fgets,fputs,printf) и множество других...
Цель данного раздела - краткое ознакомление с основами синтаксиса языка PHP. Более подробную информацию по конкретным составляющим синтаксиса PHP вы найдете в соответствующих разделах.
PHP и HTML
Синтаксис любого языка программирования гораздо легче "почувствовать" на примерах, нежели используя какие-то диаграммы и схемы. Поэтому приведем пример простейшего скрипта на PHP:

```
<html>
<head>
<title>Пример</title>
</head>
<body>

<?
```

```
echo "Привет, я - скрипт PHP!";
?>

</body>
</html>
```

Вы уже наверняка заметили, что это классический скрипт, с которого начинают изучение языка программирования.
Обратите внимание, что HTML-код корректно обрабатывается интерпретатором PHP.
Начало сценария вас может озадачить: разве это сценарий? Откуда HTML-тэги <html> и <body>? Вот тут-то и кроется главная особенность (кстати, чрезвычайно удобная) языка PHP: PHP-скрипт может вообще не отличаться от обычного HTML-документа.
Идем дальше. Вы, наверное, догадались, что сам код сценария начинается после открывающего тэга **<?** и заканчивается закрывающим **?>**. Итак, между этими двумя тэгами текст интерпретируется как программа, и в HTML-документ не попадает. Если же программе нужно что-то вывести, она должна воспользоваться оператором echo.
Итак, PHP устроен так, что любой текст, который расположен вне программных блоков, ограниченных **<?** и **?>**, выводится в браузер непосредственно. В этом и заключается главная особенность PHP, в отличие от Perl и C, где вывод осуществляется только с помощью стандартных операторов.

Разделение инструкций

Инструкции разделяются также как и в C или Perl - каждое выражение заканчивается точкой с запятой.
Закрывающий тег (?>) также подразумевает конец инструкции, поэтому два следующих фрагмента кода эквиваленты:

```
<?php
    echo "Это тест";
?>

<?php echo "Это тест" ?>
```

Комментарии в PHP скриптах

Написание практически любого скрипта не обходится без комментариев.
PHP поддерживает комметарии в стиле 'C', 'C++' и оболочки Unix. Например:

```
<?php
    echo "Это тест"; // Это однострочный комментарий в стиле c++
    /* Это многострочный комментарий
       еще одна строка комментария */
    echo "Это еще один тест";
    echo "Последний тест"; # Это комментарий в стиле оболочки Unix
?>
```

Однострочные комментарии идут только до конца строки или текущего блока PHP-кода, в зависимости от того, что идет перед ними.

```
<h1>Это <?php # echo "простой";?> пример.</h1>
<p>Заголовок вверху выведет 'Это пример'.
```

Будьте внимательны, следите за отсутствием вложенных 'C'-комментариев, они могут появиться во время комментирования больших блоков:

```
<?php
    /*
```

```
echo "Это тест"; /* Этот комментарий вызовет проблему */
*/
?>
```

Однострочные комментарии идут только до конца строки или текущего блока PHP-кода, в зависимости от того, что идет перед ними. Это означает, что HTML-код после // ?> БУДЕТ напечатан: ?> выводит из режима PHP и возвращает в режим HTML, но // не позволяет этого сделать.

Переменные в PHP

Имена переменных обозначаются знаком **$**. То же самое "Привет, я - скрипт PHP! " можно получить следующим образом:

```
<?php
$message = "Привет, я - скрипт PHP!";
echo $message;
?>
```

Подробно о переменных в PHP **здесь**

Типы данных в PHP

PHP поддерживает восемь простых типов данных:

Четыре скалярных типа:

- boolean (двоичные данные)
- integer (целые числа)
- float (числа с плавающей точкой или 'double')
- string (строки)

Два смешанных типа:

- array (массивы)
- object (объекты)

И два специальных типа:

resource (ресурсы)

NULL ("пустые")

Существуют также несколько псевдотипов:

- mixed (смешанные)
- number (числа)
- callback (обратного вызова)

Подробно о типах данных в PHP **здесь**

Выражения в PHP

Основными формами выражений являются константы и переменные. Например, если вы записываете "$a = 100", вы присваиваете '100' переменной $a:

```
$a = 100;
```

В приведенном примере $a - это переменная, = - это оператор присваивания, а 100 - это и есть выражения. Его значение 100.

Выражением может быть и переменная, если ей сопоставлено определенное значение:

```
$x = 7;
$y = $x;
```

В первой строке рассмотренного примера выражением является константа 7, а во второй строке - переменная $x, т.к. ранее ей было присвоено значение 7. $y = $x также является выражением.

Подробно о выражениях в PHP вы найдете **здесь**

Операторы PHP

Оператором называется нечто, состоящее из одного или более значений (выражений, если говорить на жаргоне программирования), которое можно вычислить как новое значение (таким образом, вся конструкция может рассматриваться как выражение).
Примеры операторов PHP:
Операторы присвоения:

```
<?php

$a = ($b = 4) + 5; // результат: $a установлена значением 9, переменной $b присвоено 4.

?>
```

Комбинированные операторы:

```
<?php

$a = 3;
$a += 5; // устанавливает $a значением 8, аналогично записи: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // устанавливает $b строкой "Hello There!", как и $b = $b . "There!";

?>
```

Строковые операторы:

```
<?php
$a = "Hello ";
$b = $a . "World!"; // $b содержит строку "Hello World!"

$a = "Hello ";
$a .= "World!"; // $a содержит строку "Hello World!"
?>
```

Существуют также логические операторы и операторы сравнения, однако их принято рассматривать в контексте управляющих конструкций языка.
Подробную информацию по операторам PHP вы найдете **здесь.**

Управляющие конструкции языка PHP

Основными конструкциями языка PHP являются:

1. Условные операторы (if, else);
2. Циклы (while, do-while, for, foreach, break, continue);
3. Конструкции выбора (switch);
4. Конструкции объявления (declare);
5. Конструкции возврата значений (return);
6. Конструкции включений (require, include).

Примеры конструкций языка PHP:

```
<?php
if ($a > $b) echo "значение a больше, чем b";
?>
```

Приведенный пример наглядно показывает использование конструкции **if** совместно с оператором сравнения ($a > $b).
В следующем примере если переменная *$a* не равна нулю, будет выведена строка "значение a истинно (true), то есть показано взаимодействие условного оператора (конструкции) if с логическим оператором:

```
<?php
if ($a) echo "значение a истинно (true) ";
?>
```

А вот пример цикла while:

```
<?php
$x=0;
while ($x++<10) echo $x;
// Выводит 12345678910
?>
```

Информацию по всем управляющим конструкциям PHP вы можете получить **здесь**

Пользовательские функции в PHP

В любом языке программирования существуют подпрограммы. В языке C они называются функциями, в ассемблере - подпрограммами, а в Pascal существуют два вида подпрограмм: процедуры и функции.

В PHP такими подпрограммами являются пользовательские функции.

Подпрограмма - это специальным образом оформленный фрагмент программы, к которому можно обратиться из любого места внутри программы. Подпрограммы существенно упрощают жизнь программистам, улучшая читабельность исходного кода, а также сокращая его, поскольку отдельные фрагменты кода не нужно писать несколько раз.

Приведем пример пользовательской функции на PHP:

```
<?php

function funct() {
$a = 100;
echo "<h4>$a</h4>";
}
funct();

?>
```

Сценарий выводит 100:

```
100
```

Пользовательским функциям в PHP можно передавать аргументы и получать возвращаемые функциями значения.

Подробную информацию по пользовательским функциям PHP вы найдете **здесь**

Встроенные (стандартные) функции PHP

PHP содержит огромное колличество встроенных функций, способных выполнять задачи различного уровня сложности.

Портал PHP.SU содержит полный справочник по стандартным функциям PHP.

ООП и PHP

PHP имеет достаточно хорошую поддержку объектно-ориентированного программирования (ООП).

В PHP можно создавать классы различных уровней, объекты и достаточно гибко ими оперировать.

Вот пример PHP класса и его использования:

```
<?php
// Создаем новый класс Coor:
```

```
class Coor {
// данные (свойства):
var $name;

// методы:
function Getname() {
echo "<h3>John</h3>";
}

}

// Создаем объект класса Coor:
$object = new Coor;
// Получаем доступ к членам класса:
$object->name = "Alex";
echo $object->name;
// Выводит 'Alex'
// А теперь получим доступ к методу класса (фактически, к функции внутри класса):
$object->Getname();
// Выводит 'John' крупными буквами
?>
```

Практическая работа №36
Операторы языка программирования PHP. Операторы присваивания, арифметические операторы, операторы отношения, логические операторы, поразрядные операторы, строковые операторы. Управляющие конструкции языка PHP.

Операторы PHP. Общие сведения

Для осуществления операций с переменными существуют различные группы операторов.

Оператором называется нечто, состоящее из одного или более значений (выражений, если говорить на жаргоне программирования), которое можно вычислить как новое значение (таким образом, вся конструкция может рассматриваться как выражение). Отсюда следует, что функции или любые другие конструкции, которые возвращают значение (например, **print()**) являются операторами, в отличие от всех остальных языковых конструкций (например, **echo()**), которые ничего не возвращают.

Вы можете ознакомиться со всеми операторами PHP далее.

Арифметические операторы PHP

Помните школьные основы арифметики? Описанные ниже операторы PHP работают таким же образом.

Пример	Название	Результат
-$a	Отрицание	Смена знака $a.

Пример	Название	Результат
$a + $b	Сложение	Сумма $a и $b.
$a - $b	Вычитание	Разность $a и $b.
$a * $b	Умножение	Произведение $a и $b.
$a / $b	Деление	Частное от деления $a на $b.
$a % $b	Деление по модулю	Целочисленный остаток от деления $a на $b.
$a ** $b	Возведение в степень	Результат $a в степени $b. (появилось в PHP 5.6.0)

Операция деления ("/") всегда возвращает вещественный тип, даже если оба значения были целочисленными (или строками, которые преобразуются в целые числа). В противном случае результат будет дробным.

Операция вычисления остатка от деления "**%**" работает только с целыми числами, так что применение ее к дробным может привести к нежелательному результату. Остаток *$a % $b* будет негативным, для негативных значений *$a*.

Возможно использование скобок. Приоритет одних математических операций над другими и изменение приоритетов при использовании скобок в арифметических выражениях соответствуют обычным математическим правилам.

Логические операторы PHP

Логические операторы предназначены исключительно для работы с логическими выражениями и возвращают **false**или **true.**

Приведем таблицу логических операторов PHP:

Пример	Название	Результат
$a and $b	Логическое 'и'	**TRUE** если и $a, и $b **TRUE.**
$a or $b	Логическое 'или'	**TRUE** если или $a, или $b **TRUE.**
$a xor $b	Исключающее 'или'	**TRUE** если $a, или $b **TRUE**, но не оба.
! $a	Отрицание	**TRUE** если $a не **TRUE.**
$a && $b	Логическое 'и'	**TRUE** если и $a, и $b **TRUE.**
$a || $b	Логическое 'или'	**TRUE** если или $a, или $b **TRUE.**

Смысл двух разных вариантов для операторов "and" и "or" в том, что они работают с различными приоритетами.

Следует заметить, что вычисление логических выражений, содержащих такие операторы, идет всегда слева направо, при этом, если результат уже очевиден (например, **false && что-то** всегда дает **false**), то вычисления обрываются, даже если в выражении присутствуют вызовы функций. Например, в операторе $logic = 0&&(time()>100); стандартная функция **time()** никогда не будет вызвана.

Будьте осторожны с логическими операциями — не забывайте про удваивание символа. Обратите внимание, что, например, | и || — два совершенно разных оператора, один из которых может потенциально возвращать любое число, а второй — только **false** и **true.**

Операторы инкремента (++) и декремента (--) не работают с логическими переменными.

Строковые операторы PHP

В PHP есть два оператора для работы со строками. Первый - оператор конкатенации ('.'), который возвращает объединение левого и правого аргумента. Второй - оператор присвоения с конкатенацией, который присоединяет правый аргумент к левому. Приведем конкретный пример:

```
<?php
$a = "Hello ";
$b = $a . "World!"; // $b содержит строку "Hello World!"

$a = "Hello ";
$a .= "World!"; // $a содержит строку "Hello World!"
?>
```

Побитовые операторы PHP

Побитовые операции предназначены для работы (установки/снятия/проверки) групп битов в целой переменной. Биты целого числа — это не что иное, как отдельные разряды того же самого числа, записанного в двоичной системе счисления. Например, в двоичной системе число 12 будет выглядеть как 1100, а 2 — как 10, так что выражение **12|2** вернет нам число 14 (1110 в двоичной записи). Если переменная не целая, то она вначале округляется, а уж затем к ней применяются перечисленные ниже операторы.

Для представления одного числа используются 32 бита:

- 0000 0000 0000 0000 0000 0000 0000 0000 - это ноль;
- 0000 0000 0000 0000 0000 0000 0000 0001 - это 1;
- 0000 0000 0000 0000 0000 0000 0000 0010 - это 2;
- 0000 0000 0000 0000 0000 0000 0000 0011 - это 3;
- 0000 0000 0000 0000 0000 0000 0000 0100 - это 4;
- 0000 0000 0000 0000 0000 0000 0000 0101 - это 5;
- ...
- 0000 0000 0000 0000 0000 0000 0000 1111 - это 15;
- ...

Побитовые операторы:

Пример	**Название**	**Результат**
$a & $b	Побитовое 'и'	Устанавливаются только те биты, которые установлены и в $a, и в $b.

Пример	Название	Результат
$a | $b	Побитовое 'или'	Устанавливаются те биты, которые установлены либо в $a, либо в $b.
$a ^ $b	Исключающее или	Устанавливаются только те биты, которые установлены либо только в $a, либо только в $b
~ $a	Отрицание	Устанавливаются те биты, которые в $a не установлены, и наоборот.
$a << $b	Сдвиг влево	Все биты переменной $a сдвигаются на $b позиций влево (каждая позиция подразумевает 'умножение на 2')
$a >> $b	Сдвиг вправо	Все биты переменной $a сдвигаются на $b позиций вправо (каждая позиция подразумевает 'деление на 2')

В случае если присутствуют и левый, и правый операнды строки, побитовые операции будут работать с их ASCII-представлениями. Пример:

```
<?php
echo 12 ^ 9; // Выведет '5'

echo "12" ^ "9"; // Отобразит симовол возврата каретки (ascii 8)
                 // ('1' (ascii 49)) ^ ('9' (ascii 57)) = #8

echo "hallo" ^ "hello"; // Выведет следующие ASCII-значения: #0 #4 #0 #0 #0
                        // 'a' ^ 'e' = #4
?>
```

Примечание: Не используйте сдвиг вправо более чем на 32 бита на тридцатидвухразрядных системах. Не используйте сдвиг вправо для получения чисел, требующих для записи более 32-х бит.

Операторы присвоения в PHP

Базовый оператор присвоения обозначается как =. На первый взгляд может показаться, что это оператор "равно". На самом деле это не так. В действительности, оператор присвоения означает, что левый операнд получает значение правого выражения, (т.е. устанавливается результирующим значением).

Результатом выполнения оператора присвоения является само присвоенное значение. Таким образом, результат выполнения *$a = 3* будет равен *3*. Это позволяет использовать конструкции вида:

```
<?php

$a = ($b = 4) + 5; // результат: $a установлена значением 9, переменной $b присвоено 4.

?>
```

В дополнение к базовому оператору присвоения имеются "комбинированные операторы" для всех бинарных арифметических и строковых операций, которые позволяют использовать некоторое значение в выражении, а затем установить его как результат данного выражения. Например:

```
<?php

$a = 3;
$a += 5; // устанавливает $a значением 8, аналогично записи: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // устанавливает $b строкой "Hello There!", как и $b = $b . "There!";

?>
```

Обратите внимание, что присвоение копирует оригинальную переменную в новую (присвоение по значению), таким образом все последующие изменения одной из переменных на другой никак не отражаются. Начиная с PHP 4, также поддерживается присваивание по ссылке, используя синтаксис $var = &$othervar; 'Присвоение по ссылке' означает, что обе переменные указывают на одни и те же данные и никакого копирования не происходит.

Практическая работа №37
Использование функций в PHP.

Функции, определяемые пользователем ¶

Приведем пример синтаксиса, используемого для описания функций:

Пример #1 Псевдокод для демонстрации использования функций

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
    echo "Example function.\n";
    return $retval;
}
?>
```

Внутри функции можно использовать любой корректный PHP-код, в том числе другие функции и даже объявления классов.

Имена функций следуют тем же правилам, что и другие метки в PHP. Корректное имя функции начинается с буквы или знака подчеркивания, за которым следует любое количество букв, цифр или знаков подчеркивания. В качестве регулярного выражения оно может быть выражено так: *[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]**.

Подсказка

Смотрите также Руководство по именованию.

Функции не обязаны быть определены до их использования, *исключая* тот случай, когда функции определяются условно, как это показано в двух последующих примерах.

В случае, когда функция определяется в зависимости от какого-либо условия, например, как это показано в двух приведенных ниже примерах, обработка описания функции должна*предшествовать* ее вызову.

Пример #2 Функции, зависящие от условий

```
<?php

$makefoo = true;

/* Мы не можем вызвать функцию foo() в этом месте,
   поскольку она еще не определена, но мы можем
   обратиться к bar() */

bar();

if ($makefoo) {
  function foo()
  {
    echo "Я не существую до тех пор, пока выполнение программы меня не достигнет.\n";
  }
}

/* Теперь мы благополучно можем вызывать foo(),
   поскольку $makefoo была интерпретирована как true */

if ($makefoo) foo();

function bar()
{
    echo "Я существую сразу с начала старта программы.\n";
}

?>
```

Пример #3 Вложенные функции

```
<?php
function foo()
{
  function bar()
  {
    echo "Я не существую пока не будет вызвана foo().\n";
  }
```

```
}

/* Мы пока не можем обратиться к bar(),
   поскольку она еще не определена. */

foo();

/* Теперь мы можем вызвать функцию bar(),
   обработка foo() сделала ее доступной. */

bar();

?>
```

Все функции и классы PHP имеют глобальную область видимости - они могут быть вызваны вне функции, даже если были определены внутри и наоборот.

PHP не поддерживает перегрузку функции, также отсутствует возможность переопределить или удалить объявленную ранее функцию.

Замечание: Имена функций регистронезависимы, тем не менее, более предпочтительно вызывать функции так, как они были объявлены.

Функции PHP поддерживают как списки аргументов переменной длины, так и значения аргументов по умолчанию. Смотрите также описания функций func_num_args(),func_get_arg(), и func_get_args() для более детальной информации.

Можно вызывать функции PHP рекурсивно.

Пример #4 Рекурсивные функции

```
<?php
function recursion($a)
{
    if ($a < 20) {
        echo "$a\n";
        recursion($a + 1);
    }
}
?>
```

Замечание: Рекурсивный вызов методов/процедур с глубиной более 100-200 уровней рекурсии может вызвать переполнение стека и привести к аварийному завершению скрипта. В частности, бесконечная рекурсия будет считаться программной ошибкой.

Практическая работа №38
Работа с массивами данных в PHP.

На самом деле массив в PHP - это упорядоченное отображение, которое устанавливает соответствие между *значением* и *ключом*. Этот тип оптимизирован в нескольких направлениях, поэтому вы можете использовать его как собственно массив, список (вектор), хэш-таблицу (являющуюся реализацией карты), словарь, коллекцию, стэк, очередь и, возможно, что-то еще. Так как значением массива может быть другой массив PHP, можно также создавать деревья и многомерные массивы.

Объяснение этих структур данных выходит за рамки данного справочного руководства, но вы найдете как минимум один пример по каждой из них. За дополнительной информацией вы можете обратиться к соответствующей литературе по этой обширной теме.

Синтаксис ¶

Определение при помощи array() ¶

Массив (тип array) может быть создан языковой конструкцией array(). В качестве параметров она принимает любое количество разделенных запятыми пар *key => value* (ключ => значение).

```
array(
key  => value,
key2 => value2,
key3 => value3,
...
)
```

Запятая после последнего элемента массива необязательна и может быть опущена. Обычно это делается для однострочных массивов, т.е. *array(1, 2)* предпочтительней *array(1, 2,)*. Для многострочных массивов с другой стороны обычно используется завершающая запятая, так как позволяет легче добавлять новые элементы в конец массива.

Начиная с PHP 5.4 возможно использовать короткий синтаксис определения массивов, который заменяет языковую конструкцию *array()* на *[]*.

Пример #1 Простой массив

```
<?php
$array = array(
    "foo" => "bar",
    "bar" => "foo",
);

// Начиная с PHP 5.4
$array = [
    "foo" => "bar",
    "bar" => "foo",
];
?>
```

key может быть либо типа integer, либо типа string. value может быть любого типа.

Дополнительно с ключом key будут сделаны следующие преобразования:

- Строки, содержащие целое число (исключая случаи, когда число предваряется знаком +) будут преобразованы к типу integer. Например, ключ со значением *"8"* будет в действительности сохранен со значением *8*. С другой стороны, значение *"08"* не будет преобразовано, так как оно не является корректным десятичным целым.
- Числа с плавающей точкой (тип float) также будут преобразованы к типу integer, т.е. дробная часть будет отброшена. Например, ключ со значением *8.7* будет в действительности сохранен со значением *8*.
- Тип bool также преобразовываются к типу integer. Например, ключ со значением *true*будет сохранен со значением *1* и ключ со значением *false* будет сохранен со значением *0*.
- Тип null будет преобразован к пустой строке. Например, ключ со значением *null* будет в действительности сохранен со значением *""*.
- Массивы (тип array) и объекты (тип object) *не могут* использоваться в качестве ключей. При подобном использовании будет генерироваться предупреждение: *Недопустимый тип смещения (Illegal offset type)*.

Если несколько элементов в объявлении массива используют одинаковый ключ, то только последний будет использоваться, а все другие будут перезаписаны.

Пример #2 Пример преобразования типов и перезаписи элементов

```
<?php
$array = array(
    1    => "a",
    "1"  => "b",
    1.5  => "c",
    true => "d",
);
var_dump($array);
?>
```

Результат выполнения данного примера:

```
array(1) {
  [1]=>
  string(1) "d"
}
```

Так как все ключи в вышеприведенном примере преобразуются к *1*, значение будет перезаписано на каждый новый элемент и останется только последнее присвоенное значение *"d"*.

Массивы в PHP могут содержать ключи типов integer и string одновременно, так как PHP не делает различия между индексированными и ассоциативными массивами.

Пример #3 Смешанные ключи типов integer и string

```
<?php
$array = array(
    "foo" => "bar",
    "bar" => "foo",
    100 => -100,
    -100 => 100,
);
var_dump($array);
?>
```

Результат выполнения данного примера:

```
array(4) {
  ["foo"]=>
  string(3) "bar"
  ["bar"]=>
  string(3) "foo"
  [100]=>
  int(-100)
  [-100]=>
  int(100)
}
```

Параметр key является необязательным. Если он не указан, PHP будет использовать предыдущее наибольшее значение ключа типа integer, увеличенное на 1.

Пример #4 Индексированные массивы без ключа

```
<?php
$array = array("foo", "bar", "hallo", "world");
var_dump($array);
?>
```

Результат выполнения данного примера:

```
array(4) {
  [0]=>
  string(3) "foo"
  [1]=>
  string(3) "bar"
  [2]=>
  string(5) "hallo"
  [3]=>
  string(5) "world"
}
```

Возможно указать ключ только для некоторых элементов и пропустить для других:

Пример #5 Ключи для некоторых элементов

```
<?php
$array = array(
    "a",
    "b",
    6 => "c",
    "d",
);
var_dump($array);
?>
```

Результат выполнения данного примера:

```
array(4) {
  [0]=>
  string(1) "a"
  [1]=>
  string(1) "b"
  [6]=>
  string(1) "c"
  [7]=>
  string(1) "d"
}
```

Как вы видите последнее значение *"d"* было присвоено ключу *7*. Это произошло потому, что самое большое значение ключа целого типа перед этим было *6*.

Доступ к элементам массива с помощью квадратных скобок ¶

Доступ к элементам массива может быть осуществлен с помощью синтаксиса *array[key]*.

Пример #6 Доступ к элементам массива

```
<?php
$array = array(
    "foo" => "bar",
    42 => 24,
    "multi" => array(
        "dimensional" => array(
            "array" => "foo"
        )
    )
);

var_dump($array["foo"]);
var_dump($array[42]);
var_dump($array["multi"]["dimensional"]["array"]);
?>
```

Результат выполнения данного примера:

```
string(3) "bar"
int(24)
string(3) "foo"
```

Замечание:
И квадратные и фигурные скобки можно взаимозаменяемо использовать для доступа к элементам массива (т.е. и *$array[42]* и *$array{42}* равнозначны).

С PHP 5.4 стало возможным прямое разыменование массива, возвращаемого в качестве результата вызова функции или метода. Раньше приходилось использовать временные переменные.

С PHP 5.5 возможно разыменовывать массив буквально.

Пример #7 Разыменование массива

```
<?php
function getArray() {
    return array(1, 2, 3);
}

// в PHP 5.4
$secondElement = getArray()[1];

// ранее делали так
$tmp = getArray();
$secondElement = $tmp[1];

// или так
list(, $secondElement) = getArray();
?>
```

Замечание:
Попытка доступа к неопределенному ключу в массиве - это то же самое, что и попытка доступа к любой другой неопределенной переменной: будет сгенерирована ошибка уровня **E_NOTICE**, и результат будет **NULL**.

Замечание:
Массив разыменовывающий скалярную величину не являющуюся строкой (string), отдаст **NULL** без какого либо оповещения об ошибке.

Создание/модификация с помощью синтаксиса квадратных скобок ¶

Существующий массив может быть изменен явной установкой значений в нем.

Это выполняется присвоением значений массиву array с указанием в скобках ключа. Кроме того, вы можете опустить ключ. В этом случае добавьте к имени переменной пустую пару скобок (*[]*).

```
$arr[key] = value;
```

```
$arr[] = value;
// key может быть integer или string
// value может быть любым значением любого типа
```

Если массив $arr еще не существует, он будет создан. Таким образом, это еще один способ определить массив array. Однако такой способ применять не рекомендуется, так как если переменная $arr уже содержит некоторое значение (например, значение типа string из переменной запроса), то это значение останется на месте и *[]* может на самом деле означатьдоступ к символу в строке. Лучше инициализировать переменную путем явного присваивания значения.

Замечание: Начиная с PHP 7.1.0, используя в оператор "пустой индекс" на строке, приведет к фатальной ошибке. Ранее, в этом случае, строка молча преобразовывалась в массив.

Для изменения определенного значения просто присвойте новое значение элементу, используя его ключ. Если вы хотите удалить пару ключ/значение, вам необходимо использовать функцию unset().

```
<?php
$arr = array(5 => 1, 12 => 2);

$arr[] = 56;    // В этом месте скрипта это
                // то же самое, что и $arr[13] = 56;

$arr["x"] = 42; // Это добавляет к массиву новый
                // элемент с ключом "x"

unset($arr[5]); // Это удаляет элемент из массива

unset($arr);    // Это удаляет массив полностью
?>
```

Замечание:
Как уже говорилось выше, если ключ не был указан, то будет взят максимальный из существующих целочисленных (integer) индексов, и новым ключом будет это максимальное значение (в крайнем случае 0) плюс 1. Если целочисленных индексов еще нет, то ключом будет *0* (ноль).

Учтите, что максимальное целое значение ключа *не обязательно существует в массиве в данный момент*. Оно могло просто существовать в массиве какое-то время, с тех пор как он был переиндексирован в последний раз. Следующий пример это иллюстрирует:

```
<?php
// Создаем простой массив.
$array = array(1, 2, 3, 4, 5);
print_r($array);

// Теперь удаляем каждый элемент, но сам массив оставляем нетронутым:
foreach ($array as $i => $value) {
    unset($array[$i]);
```

```
}
print_r($array);

// Добавляем элемент (обратите внимание, что новым ключом будет 5, вместо 0).
$array[] = 6;
print_r($array);

// Переиндексация:
$array = array_values($array);
$array[] = 7;
print_r($array);
?>
```

Результат выполнения данного примера:

```
Array
(
   [0] => 1
   [1] => 2
   [2] => 3
   [3] => 4
   [4] => 5
)
Array
(
)
Array
(
   [5] => 6
)
Array
(
   [0] => 6
   [1] => 7
)
```

Полезные функции ¶

Для работы с массивами существует достаточное количество полезных функций. Смотрите раздел функции для работы с массивами.

Замечание:
Функция unset() позволяет удалять ключи массива. Обратите внимание, что массив *НЕ*будет переиндексирован. Если вы действительно хотите поведения в стиле "удалить и сдвинуть", можно переиндексировать массив используя array_values().

```
<?php
$a = array(1 => 'один', 2 => 'два', 3 => 'три');
unset($a[2]);
/* даст массив, представленный так:
   $a = array(1 => 'один', 3 => 'три');
   а НЕ так:
```

```
                $a = array(1 => 'один', 2 => 'три');
*/

$b = array_values($a);
// Теперь $b это array(0 => 'один', 1 => 'три')
?>
```

Управляющая конструкция foreach существует специально для массивов. Она предоставляет возможность легко пройтись по массиву.

Что можно и нельзя делать с массивами ¶

Почему *$foo[bar]* неверно? ¶

Всегда заключайте в кавычки строковый литерал в индексе ассоциативного массива. К примеру, пишите *$foo['bar']*, а не *$foo[bar]*. Но почему? Часто в старых скриптах можно встретить следующий синтаксис:

```
<?php
$foo[bar] = 'враг';
echo $foo[bar];
// и т.д.
?>
```

Это неверно, хотя и работает. Причина в том, что этот код содержит неопределенную константу (*bar*), а не строку (*'bar'* - обратите внимание на кавычки). Это работает, потому что PHP автоматически преобразует *"голую строку"* (не заключенную в кавычки строку, которая не соответствует ни одному из известных символов языка) в строку со значением этой "голой строки". Например, если константа с именем **bar** не определена, то PHP заменит bar на строку *'bar'* и использует ее.

Замечание: Это не означает, что нужно *всегда* заключать ключ в кавычки. Нет необходимости заключать в кавычки константы или переменные, поскольку это помешает PHP обрабатывать их.

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', true);
ini_set('html_errors', false);
// Простой массив:
$array = array(1, 2);
$count = count($array);
for ($i = 0; $i < $count; $i++) {
    echo "\nПроверяем $i: \n";
    echo "Плохо: " . $array['$i'] . "\n";
    echo "Хорошо: " . $array[$i] . "\n";
    echo "Плохо: {$array['$i']}\n";
    echo "Хорошо: {$array[$i]}\n";
}
?>
```

Результат выполнения данного примера:

```
Проверяем 0:
Notice: Undefined index:  $i in /path/to/script.html on line 9
Плохо:
Хорошо: 1
Notice: Undefined index:  $i in /path/to/script.html on line 11
Плохо:
Хорошо: 1

Проверяем 1:
Notice: Undefined index:  $i in /path/to/script.html on line 9
Плохо:
Хорошо: 2
Notice: Undefined index:  $i in /path/to/script.html on line 11
Плохо:
Хорошо: 2
```

Дополнительные примеры, демонстрирующие этот факт:

```
<?php
// Показываем все ошибки
error_reporting(E_ALL);

$arr = array('fruit' => 'apple', 'veggie' => 'carrot');

// Верно
print $arr['fruit']; // apple
print $arr['veggie']; // carrot

// Неверно. Это работает, но из-за неопределенной константы с
// именем fruit также вызывает ошибку PHP уровня E_NOTICE
//
// Notice: Use of undefined constant fruit - assumed 'fruit' in...
print $arr[fruit]; // apple

// Давайте определим константу, чтобы продемонстрировать, что
// происходит. Мы присвоим константе с именем fruit значение 'veggie'.
define('fruit', 'veggie');

// Теперь обратите внимание на разницу
print $arr['fruit']; // apple
print $arr[fruit]; // carrot

// Внутри строки это нормально. Внутри строк константы не
// рассматриваются, так что ошибки E_NOTICE здесь не произойдет
print "Hello $arr[fruit]"; // Hello apple

// С одним исключением: фигурные скобки вокруг массивов внутри
```

```
// строк позволяют константам там находиться
print "Hello {$arr[fruit]}"; // Hello carrot
print "Hello {$arr['fruit']}"; // Hello apple

// Это не будет работать и вызовет ошибку обработки, такую как:
// Parse error: parse error, expecting T_STRING' or T_VARIABLE' or T_NUM_STRING'
// Это, конечно, также действует и с суперглобальными переменными в строках
print "Hello $arr['fruit']";
print "Hello $_GET['foo']";

// Еще одна возможность - конкатенация
print "Hello " . $arr['fruit']; // Hello apple
?>
```

Если вы переведете error_reporting в режим отображения ошибок уровня **E_NOTICE**(например, такой как **E_ALL**), вы сразу увидите эти ошибки. По умолчанию error_reportingустановлена их не отображать.

Как указано в разделе синтаксис, внутри квадратных скобок ('*[*' и '*]*') должно быть выражение. Это означает, что можно писать вот так:

```
<?php
echo $arr[somefunc($bar)];
?>
```

Это пример использования возвращаемого функцией значения в качестве индекса массива. PHP известны также и константы:

```
<?php
$error_descriptions[E_ERROR] = "Произошла фатальная ошибка";
$error_descriptions[E_WARNING] = "PHP сообщает о предупреждении";
$error_descriptions[E_NOTICE] = "Это лишь неофициальное замечание";
?>
```

Обратите внимание, что **E_ERROR** - это такой же верный идентификатор, как и *bar* в первом примере. Но последний пример по сути эквивалентен такой записи:

```
<?php
$error_descriptions[1] = "Произошла фатальная ошибка";
$error_descriptions[2] = "PHP сообщает о предупреждении";
$error_descriptions[8] = "Это лишь неофициальное замечание";
?>
```

поскольку **E_ERROR** соответствует *1*, и т.д.

Так что же в этом плохого?

Когда-нибудь в будущем, команда разработчиков PHP, возможно, пожелает добавить еще одну константу или ключевое слово, либо константа из другого кода может вмешаться и

тогда у вас могут возникнуть проблемы. Например, вы уже не можете использовать таким образом слова *empty* и *default*, поскольку они являются зарезервированными ключевыми словами.

Замечание: Повторим, внутри строки (string), заключенной в двойные кавычки, корректно не окружать индексы массива кавычками, поэтому *"$foo[bar]"* является верной записью. Более подробно почему - смотрите вышеприведенные примеры, а также разделобработка переменных в строках.

Преобразование в массив ¶

Для любого из типов integer, float, string, boolean и resource, преобразование значения в массив дает результатом массив с одним элементом (с индексом 0), являющимся скалярным значением, с которого вы начали. Другими словами, *(array)$scalarValue* - это точно то же самое, что и *array($scalarValue)*.

Если вы преобразуете в массив объект (object), вы получите в качестве элементов массива свойства (переменные-члены) этого объекта. Ключами будут имена переменных-членов, с некоторыми примечательными исключениями: целочисленные свойства станут недоступны; к закрытым полям класса (private) спереди будет дописано имя класса; к защищенным полям класса (protected) спереди будет добавлен символ '*'. Эти добавленные значения с обоих сторон также имеют нулевые байты. Это может вызвать несколько неожиданное поведение:

```
<?php

class A {
    private $A; // Это станет '\0A\0A'
}

class B extends A {
    private $A; // Это станет '\0B\0A'
    public $AA; // Это станет 'AA'
}

var_dump((array) new B());
?>
```

Вышеприведенный код покажет 2 ключа с именем 'AA', хотя один из них на самом деле имеет имя '\0A\0A'.

Если вы преобразуете в массив значение **NULL**, вы получите пустой массив.

Сравнение ¶

Массивы можно сравнивать при помощи функции array_diff() и операторов массивов.

Примеры ¶

Тип массив в PHP является очень гибким, вот несколько примеров:

```
<?php
// это
$a = array( 'color' => 'red',
'taste' => 'sweet',
'shape' => 'round',
'name' => 'apple',
4 // ключом будет 0
);

$b = array('a', 'b', 'c');

// . . .полностью соответствует
$a = array();
$a['color'] = 'red';
$a['taste'] = 'sweet';
$a['shape'] = 'round';
$a['name'] = 'apple';
$a[] = 4; // ключом будет 0

$b = array();
$b[] = 'a';
$b[] = 'b';
$b[] = 'c';

// после выполнения этого кода, $a будет массивом
// array('color' => 'red', 'taste' => 'sweet', 'shape' => 'round',
// 'name' => 'apple', 0 => 4), a $b будет
// array(0 => 'a', 1 => 'b', 2 => 'c'), или просто array('a', 'b', 'c').
?>
```

Пример #8 Использование array()

```
<?php
// Массив как карта (свойств)
$map = array( 'version' => 4,
'OS' => 'Linux',
'lang' => 'english',
'short_tags' => true
);

// исключительно числовые ключи
$array = array( 7,
8,
0,
156,
-10
);
```

```
// это то же самое, что и array(0 => 7, 1 => 8, ...)

$switching = array( 10, // ключ = 0
                    5 => 6,
                    3 => 7,
                    'a' => 4,
                    11, // ключ = 6 (максимальным числовым индексом было 5)
                    '8' => 2, // ключ = 8 (число!)
                    '02' => 77, // ключ = '02'
                    0 => 12 // значение 10 будет перезаписано на 12
                  );

// пустой массив
$empty = array();
?>
```

Пример #9 Коллекция

```
<?php
$colors = array('red', 'blue', 'green', 'yellow');

foreach ($colors as $color) {
    echo "Вам нравится $color?\n";
}

?>
```

Результат выполнения данного примера:

```
Вам нравится red?
Вам нравится blue?
Вам нравится green?
Вам нравится yellow?
```

Изменение значений массива напрямую возможно путем передачи их по ссылке.

Пример #10 Изменение элемента в цикле

```
<?php
foreach ($colors as &$color) {
    $color = strtoupper($color);
}
unset($color); /* это нужно для того, чтобы последующие записи в $color не меняли последний элемент массива */

print_r($colors);
?>
```

Результат выполнения данного примера:

```
Array
(
   [0] => RED
   [1] => BLUE
   [2] => GREEN
   [3] => YELLOW
)
```

Следующий пример создает массив, начинающийся с единицы.

Пример #11 Индекс, начинающийся с единицы

```
<?php
$firstquarter = array(1 => 'Январь', 'Февраль', 'Март');
print_r($firstquarter);
?>
```

Результат выполнения данного примера:

```
Array
(
   [1] => 'Январь'
   [2] => 'Февраль'
   [3] => 'Март'
)
```

Пример #12 Заполнение массива

```
<?php
// заполняем массив всеми элементами из директории
$handle = opendir('.');
while (false !== ($file = readdir($handle))) {
    $files[] = $file;
}
closedir($handle);
?>
```

Массивы упорядочены. Вы можете изменять порядок элементов, используя различные функции сортировки. Для дополнительной информации смотрите раздел функции для работы с массивами. Вы можете подсчитать количество элементов в массиве с помощью функции count().

Пример #13 Сортировка массива

```
<?php
sort($files);
print_r($files);
?>
```

Поскольку значение массива может быть чем угодно, им также может быть другой массив. Таким образом вы можете создавать рекурсивные и многомерные массивы.

Пример #14 Рекурсивные и многомерные массивы

```
<?php
$fruits = array ( "фрукты" => array ( "a" => "апельсин",
                                      "b" => "банан",
                                      "c" => "яблоко"
                                    ),
                  "числа" => array ( 1,
                                     2,
                                     3,
                                     4,
                                     5,
                                     6
                                    ),
                  "дырки" => array ( "первая",
                                     5 => "вторая",
                                     "третья"
                                    )
                );

// Несколько примеров доступа к значениям предыдущего массива
echo $fruits["дырки"][5]; // напечатает "вторая"
echo $fruits["фрукты"]["a"]; // напечатает "апельсин"
unset($fruits["дырки"][0]); // удалит "первая"

// Создаст новый многомерный массив
$juices["apple"]["green"] = "good";
?>
```

Обратите внимание, что при присваивании массива всегда происходит копирование значения. Чтобы скопировать массив по ссылке, вам нужно использовать оператор ссылки.

```
<?php
$arr1 = array(2, 3);
$arr2 = $arr1;
$arr2[] = 4; // $arr2 изменился,
             // $arr1 все еще array(2, 3)

$arr3 = &$arr1;
$arr3[] = 4; // теперь $arr1 и $arr3 одинаковы
?>
```

add a note

User Contributed Notes 21 notes

Практическая работа №39
Работа с формами. Способы работы форм. Методы GET и POST. Общие принципы обработки данных из форм. Доступные элементы форм и работа с ними.

Метод GET

Метод GET использует для передачи данных строку URL. Возможно, Вы обращали внимание на длинные и непонятные URLы. Например: function.php?login=Alex&email=dezyakin. В данном случае данные обрабатываются в function.php. После знака вопроса "?" идет перечисление передаваемых параметров (параметр разделяются знаком "&") со значениями: параметру login присвоено значение Alex, а переменной email значение dezyakin. Данные будут хранится в суперглобальном массиве $_GET. Приведем пример использования метода GET представлен ниже:

```
<html>
<body>
<form method="GET"> <!--указание метода GET-->
Login: <input type="text" name="login">
E-mail: <input type="text" name="email">
<input type="submit" value="Отправить">
</form>


<?php
/*С помощью суперглобального массива $_GET
выводим принятые значения:*/
echo "<br/>login = ". $_GET['login'];
echo "<br/>email = ". $_GET['email'];
?>
</body>
</html>
```

Результат выполнения выше описанного кода представлен на рисунке ниже:

Обратите внимание на то, как мы считываем значения из суперглобального массива $_GET: $_GET['имя_переменной']. В нашем примере имена переменных были объявлены в форме (name=login и name=email).

Совет:
Прежде чем обрабатывать полученные значения советую проверять их на существование через функции isset(имя_переменной) илиempty(имя_переменной) - эти функции были рассмотрены в предыдущем уроке 2: **переменные в PHP**. Например:

```
<?php
```

```
//проверка на существование с помощью isset:
if isset($_GET['login'])
{
операторы для обработки login
...
}
//или же проверить на существование с помощью empty:
if empty($_GET['email'])
{
операторы для обработки email
...
}
?>
```

В форме можно указать имя файла, который будет обрабатывать передаваемые значения. Делается это с помощью атрибута формы action, которому можно присвоить адрес этого файла. По умолчанию этот файл присвоен текущему файлу (т.е. обрабатывается в файле, где и расположена форма). Приведем пример, в котором данные из формы передаются на обработку в файлsrcipt.php:

```
<form method="GET" action="srcipt.php">
Login: <input type="text" name="login">
E-mail: <input type="text" name="email">
<input type="submit" value="Отправить">
</form>
```

В файле script.php должен содержатся какой-то обработчик информации, иначе информация будет передана в пустую.

Метод GET обладает множеством недостатков:

- Пользователь видит значения передаваемых параметров;
- Пользователь может легко подделать передаваемые параметры;
- Неудобная передача бинарной информации (приходится кодировать в текстовый формат);
- Объем передаваемых данных ограничен - 8 Кбайт;

Из-за выше перечисленных недостатков метод GET применяется только в тех случаях, когда нужно передать небольшой объем данных, а также эти данные никак не засекречены.

Метод POST

Метод POST отличается от GET тем, что данные передаются в закрытой форме. Существует суперглобальный массив $_POST, из которого можно считывать данные следующим образом: $_POST['имя_переменной']. Например:

```
<html>
<body>
<form method="POST"> <!--указание метода POST-->
Login: <input type="text" name="login" value="<?php echo $_POST['login'];?>">
E-mail: <input type="text" name="email" value="<?php echo $_POST['email'];?>">
<input type="submit" value="Отправить">
</form>


<?php
/*С помощью суперглобального массива $_POST
выводим принятые значения:*/
echo "<br/>login = ". $_POST['login'];
echo "<br/>email = ". $_POST['email'];
?>
</body>
</html>
```

Результат выполнения выше описанного кода представлен на рисунке ниже:

Как видите URL не имеет никакой приписки, но тем не менее данные были получены и выведены.

Примечание:

1) Объем передаваемых значений методом POST по умолчанию ограничен и равен 8 Мбайт. Чтобы увеличить это значение нужно изменить директивуpost_max_size в php.ini.

2) В ранних версиях PHP вместо коротких названий суперглобальных массивов$_GET и $_POST использовались более длинные имена: $HTTP_GET_VARS и$HTTP_POST_VARS. По умолчанию они выключены в php 5, но Вы можете их включить в конфигурационном файле php.ini с помощью параметраregister_long_arrays. В php 6 версии эти длинные названия будут недоступны.

3) Перед обработкой переменных из $_POST, советую проверять переменные на их наличие, также как это делалось с методом GET.

Практическая работа №40

Работа с файлами и каталогами. Открытие, закрытие, чтение и запись, копирование, удаление и переименование файлов и каталогов.

Под работой с файлами в PHP подразумевается **чтение из файла** из**апись в файл** различной информации. Совершенно очевидно, что работать с файлами приходится много, поэтому любой **PHP-программист** обязан уметь**считывать из файла** и **записывать в файл**.

Последовательность **работы с файлами в PHP** такая:

1. Открыть файл.
2. Выполнить необходимые действия.
3. Закрыть файл.

Как видите, последовательность работы с файлами напоминает работу с файлами через обычный проводник. Только здесь вся работа выполняется автоматически самим **PHP-скриптом**.

Начнём с первого пункта - открытие файла. Файл открывается с помощью**функции fopen()**. Первый параметр - это путь к файлу, а второй параметр -**модификатор**. Давайте сразу разберём возможные модификаторы:

1. **a** - открывает файл только для записи, причём указатель помещается в конец файла.
2. **a+** - то же самое, что и модификатор **a**, но также файл открывается ещё и для чтения.
3. **r** - открывает файл только для чтения, а указатель устанавливается в начало файла.
4. **r+** - то же самое, что и модификатор **r**, но также файл открывается ещё и для записи.
5. **w** - открывает файл только для записи, указатель устанавливает в начало файла и стирает всё содержимое файла.
6. **w+** - то же самое, что и модификатор **w**, только файл открывается также и для чтения.

Также различают два режима работы с файлами: **бинарный** (обозначается **b**) и**текстовый** (обозначается **t**). Если Вы работаете с обычным текстовым файлом, то выбирайте текстовый режим, а если, например, с изображением, то бинарный.

Это все основные модификаторы, которых Вам вполне хватит. Теперь давайте узнаем, как закрыть файл. Закрывается файл с помощью **функции fclose().**

Теперь перейдём к чтению файла с помощью **функции fread()**. И давайте, наконец-то, приведу пример:

```
<?php
$handle = fopen("files/a.txt", "rt");
$contents = '';
while (!feof($handle))
$contents .= fread($handle, 4096);
fclose($handle);
?>
```

В данном примере мы сначала открываем файл для чтения в текстовом режиме (модификатор **rt**). **Функция fopen()** возвращает так называемый**дескриптор**, с помощью которого можно общаться с файлом, и записываем его в переменную **handle**. Затем мы в цикле **while()** до тех пор, пока не достигнут конец файл, считываем содержимое каждый раз по **4096** символов, которые записываем в переменную **contents**. После завершения процесса считывания - закрываем файл, вновь с помощью дескриптора файла.

Теперь перейдём к записи с помощью **функции fwrite()**:

```
<?php
$handle = fopen("files/a.txt", "at");
$string = "This is text";
fwrite($handle, $string);
```

```
fclose($handle);
?>
```

После запуска этого скрипта, в файле **a.txt** добавится строка "**This is text**".

Особо внимательные читатели обратили внимание на указатели, о которых я писал чуть выше. **Указатель** - это текущая позиция воображаемого "курсора" в файле. Именно с него и начинается работа с файлом. Изменить положение указателя можно с помощью **функции fseek()**:

```
<?php
$handle = fopen("files/a.txt", "rt");
$contents = fread($handle, 3);
echo $contents."<br />";
fseek($handle, 0, SEEK_SET);
$contents = fread($handle, 3);
echo $contents."<br />";
?>
```

Таким образом, мы сначала считываем **3** символа (в результате, текущее положение указателя сдвигается на **3** позиции). Затем мы устанавливаем указатель на начало файла. И вновь считываем **3** символа. Как Вы и догадались, мы два раза считали одно и тоже. То есть первый раз **3** символа, потом вернулись назад, и вновь считали **3** символа. Также если у **функции fseek()** заменить **SEEK_SET** на **SEEK_CUR**, то тогда второй параметр будет не устанавливать позицию указателя, а сдвигать относительно текущего местоположения. Советую даже попрактиковаться с указателями, потому что для понимания это не так просто. Также рекомендую попытаться записать что-нибудь в файл при позиции указателя, например, в самом начале файла. И обязательно объясните полученный результат.

И, напоследок, хочется привести ещё пару функций, которые позволяют работать с файлами на самом простом уровне: **file_put_contens()** и**file_get_contents()**. Функция **file_put_contents()** записывает в файл, а функция**file_get_contents()** считывает содержимое из файла. Эти функции очень просты в применении, но возможностей там уже меньше (хотя, как правило, они и не нужны):

```
<?php
file_put_contents("files/a.txt", "This is text 2");
echo file_get_contents("files/a.txt");
?>
```

В данном скрипте мы сначала записали строку "**This is text 2**" в файл, а потом считываем полученное содержимое и выводим его. Как видите, трудно придумать более простой способ **чтения из файла** и **запись в файл.**

Вы научитесь **создавать каталоги в PHP**, удалять их и считывать из них файлы и подкаталоги.

Начнём с самого простого: **создание каталога в PHP**:

```
<?php
mkdir("new_dir");
?>
```

После запуска этого скрипта у Вас будет создан пустой каталог "**new_dir**".

Удалить пустой каталог очень просто. Для этого используется **функция rmdir()**.

```
<?php
rmdir("new_dir");
?>
```

А вот теперь перейдём к **работе с содержимым каталогов через PHP**. Здесь есть очень простые правила, которые необходимо соблюдать. Все эти правила очень логичны, и Вы их применяете, когда вручную просматриваете содержимое каталогов:

1. Открыть каталог.
2. Считать содержимое.
3. Закрыть каталог.

Чтобы не мучить Вас в ожиданиях, сразу приведу код, который выводит имена файлов и категорий внутри заданного каталога:

```
<?php
$dir = opendir("images");
while (($f = readdir($dir)) !== false)
echo $f."<br />";
closedir($dir);
?>
```

В результате Вы увидите список всех файлов и каталогов внутри каталога "**images**". Также Вы увидите два интересных имени "**.**" и "**..**". Первый означает "*текущий каталог*", а ".." - родительский.

Теперь подробно о функциях, используемых в этом примере:

- Функция **opendir(string $path)** - открывает каталог, находящийся по пути**$path**, а также возвращает дескриптор, необходимый для работы с этим каталогом.
- Функция **readdir(resource $dir)** - считывает текущий элемент в каталоге**dir**. Текущий элемент задаётся указателем, который сдвигается при каждом вызове. Поэтому получается, что каждый раз эта функция возвращает новый элемент из каталога. Когда все элементы закончились, то **функция readdir()** возвращает **false**.
- Функция **closedir(resource $dir)** - закрывает каталог **dir**.

Это все самые важные **функции для работы с каталогами в PHP**. Однако, хочется добавить ещё одну очень важную деталь по поводу **функции rmdir()**, которая удаляет каталог. Если Вы внимательно читали, то я написал, что эта функция удаляет "**пустой каталог**", то есть в котором нет ни одного файла и каталога (кроме "**.**" и "**..**"). Другими словами, если в каталоге будет хотя бы один файл, то **функция rmdir() не сработает**. Вот как решить эту проблему Вы узнаете в следующей статье, поэтому подписывайтесь на обновления, чтобы не пропустить её появление.

Практическая работа №41
Работа с изображениями. Создание и вывод изображения. Рисование геометрических фигур. Работа с текстом. Примеры интерактивных изображений на странице.

Рисование точки

В этом уроке помимо стандартных способов рисования точек и линий рассматриваются более сложные - рисование сглаженной линии, рисование линий произвольной ширины. Кроме того, приведен пример частичного затемнения картинки для создании подписи.

Для рисования точки используется функцияimagesetpixel:

Код:

```
int imagesetpixel ( resource image, int x, int y, int color)
```

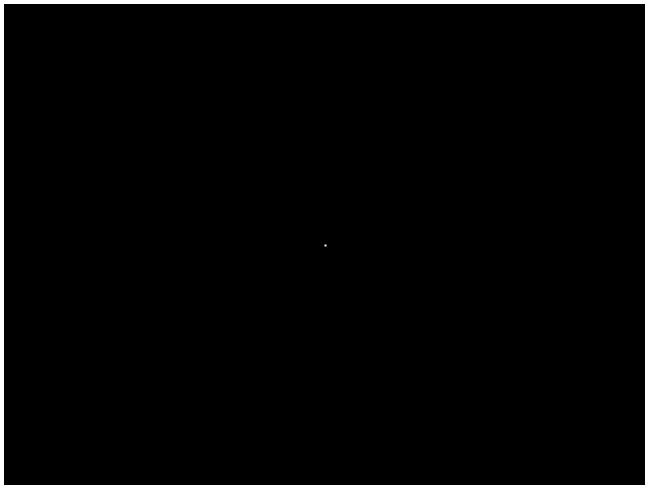
imagesetpixel() рисует на изображенииimageточку с координатамх,уи цветомcolor. Верхний левый угол имеет координаты 0,0.

Пример 7.Рисование точки

Код:

```
header ("Content-type: image/png");
$img = imagecreatetruecolor(320, 240);
$ink = imagecolorallocate($img, 255, 255, 255);
imagesetpixel($img,160,120,$ink);
imagepng($img);
imagedestroy($img);
```

Результат работы этой программы выглядит следующим образом:



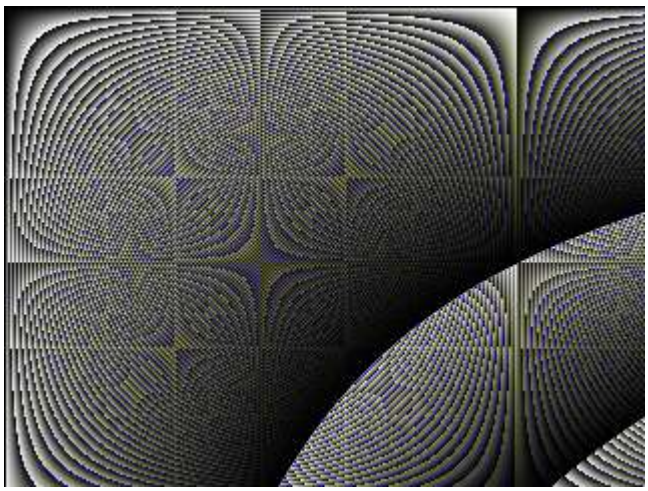
С помощью точки можно нарисовать что угодно. Для иллюстрации рассмотрим такую простую программу:

Пример 8.

Код:

```
header ("Content-type: image/png");
$img = imagecreatetruecolor(320, 240);
for ($i=0;$i<320;$i++)
    for ($j=0;$j<240;$j++) {
        $ink = imagecolorallocate($img, $i*$j, $i*$j, $i*$j);
        imagesetpixel($img,$i,$j,$ink);
    }
imagepng($img);
imagedestroy($img);
```

Результат работы этой программы выглядит следующим образом:



Рисование линии

Для рисования линии используется функцияimageline:

[code=php

int imageline (resource image, int x1, int y1, int x2, int y2, int color)

[/code]

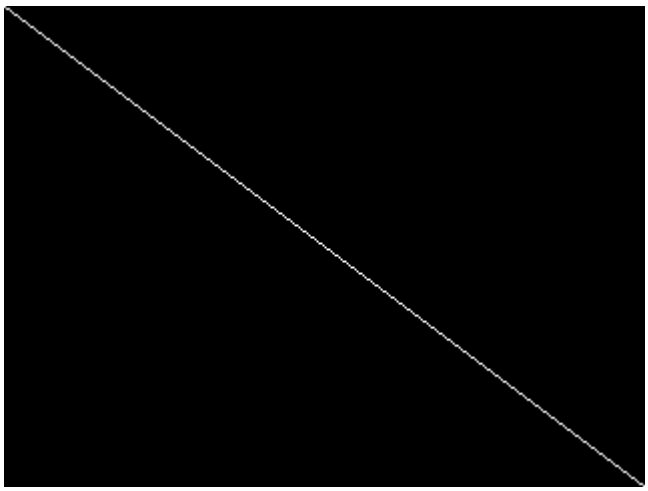
imageline() рисует на изображенииimageотрезок, начинающийся в точке x1:y1, заканчивающийся в точке x2:y2 и имеющий цветcolor. Верхний левый угол имеет координаты 0:0.

Пример 9.Рисование линии

Код:

```
header ("Content-type: image/png");
$img = imagecreatetruecolor(320, 240);
$ink = imagecolorallocate($img, 255, 255, 255);
imageline($img,0,0,320-1,240-1,$ink);
imagepng($img);
imagedestroy($img);
```

Результат работы этой программы выглядит следующим образом:



См. также:

- Вывод линии. алгоритм Брезенхема (PHP)
- @Вывод линий, алгоритм Брезенхема, алгоритм ЦДА (Pascal)
- @Вывод линии. алгоритм Брезенхема (Си)

Рисование линии произвольной ширины

Иногда появляется необходимость рисованиялинии произвольной толщины(пример будет рассмотрен в уроке 13). Функции рисования такой линии нет, но мы может сами написать ее:

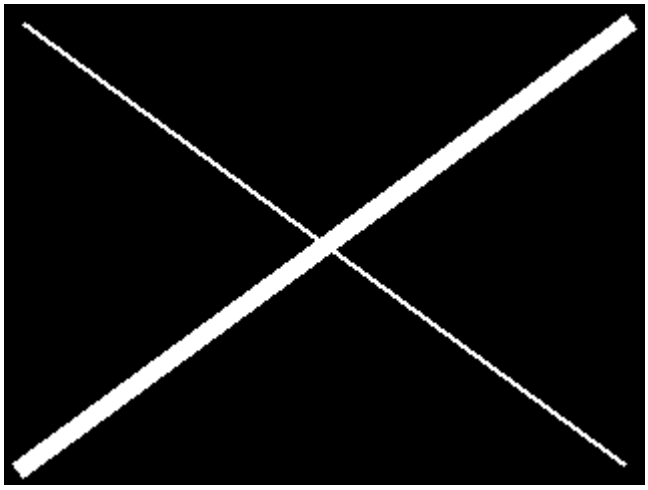
Пример 10.Рисование линии произвольной толщины.

Код:

```
function imagelinethick($image, $x1, $y1, $x2, $y2, $color, $thick = 1) {
    if ($thick == 1) {
        return imageline($image, $x1, $y1, $x2, $y2, $color);
    }
    $t = $thick / 2 - 0.5;
    if ($x1 == $x2 || $y1 == $y2) {
        return imagefilledrectangle($image, round(min($x1, $x2) - $t), round(min($y1, $y2) - $t), round(max($x1, $x2) + $t), round(max($y1, $y2) + $t), $color);
    }
    $k = ($y2 - $y1) / ($x2 - $x1); //y = kx + q
    $a = $t / sqrt(1 + pow($k, 2));
    $points = array(
        round($x1 - (1+$k)*$a), round($y1 + (1-$k)*$a),
        round($x1 - (1-$k)*$a), round($y1 - (1+$k)*$a),
        round($x2 + (1+$k)*$a), round($y2 - (1-$k)*$a),
        round($x2 + (1-$k)*$a), round($y2 + (1+$k)*$a),
    );
    imagefilledpolygon($image, $points, 4, $color);
    return imagepolygon($image, $points, 4, $color);
```

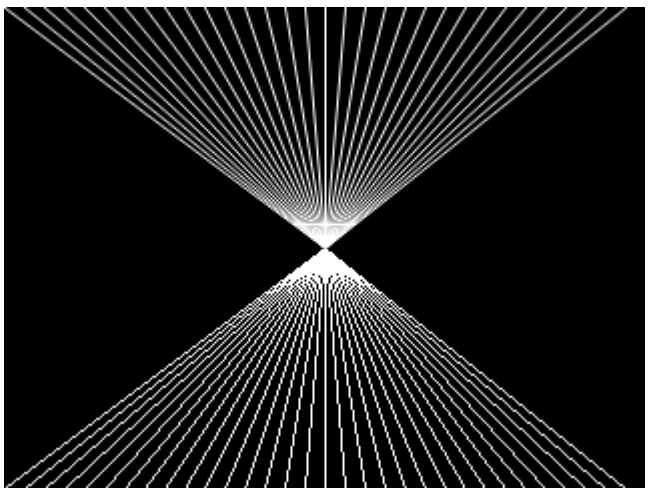
```
                                                                                }

header ("Content-type: image/png");
$img = imagecreatetruecolor(320, 240);
$ink = imagecolorallocate($img, 255, 255, 255);
imagelinethick($img,10,10,320-11,240-11,$ink,2);
imagelinethick($img,10,240-10,320-11,240-(240-11),$ink,10);
imagepng($img);
imagedestroy($img);
```

Результат работы этой программы выглядит следующим образом:



Рисование сглаженной линии

Пример рисования сглаженной линиивы можете найти здесь. Результат работы этой программы выглядит следующим образом:



На картинке хорошо видно, что линии в верхней части более гладкие. Я не рекомендую злоупотреблять функциейimagesmoothlineэтого примера. Она работает слишком медленно. Дождитесь лучше урока 13.

Заключение

В заключении хотелось бы привести пример простой программы, позволяющей выводить подпись на затемненной части изображения. Помимо функцииimagesetpixelв примере используются еще две:

- imagecolorat- возвращает значение цвета в текущей точке.
- imagecolorsforindex- возвращает ассоциативный массив с индексами "red", "green", "blue" и "alpha". В качестве параметра передается цвет, полученный с помощью функцииimagecolorat

Пример 11.Затемнение изображения

```
Код:// Коэффициент затемнения. Вычитается из каждой компоненты (R,G и B)
$DARKNESS=40;

header ("Content-type: image/png");
$img = imagecreatefromjpeg("test.jpg");

for ($i=0;$i<100;$i++)
  for ($j=0;$j<40;$j++) {
    $x=$i+530-110;
    $y=$j+10;

    $rgb = imagecolorat($img, $x, $y);
    $rgb = imagecolorsforindex($img, $rgb);

    // Если значение цвета в текущей точке больше коэффицинта затемнения
    // то просто вычитаем его. Иначе ставим в точку 0.

    $rgb["red"] = $rgb["red"] > $DARKNESS ? $rgb["red"]-$DARKNESS : 0;
    $rgb["green"] = $rgb["green"] > $DARKNESS ? $rgb["green"]-$DARKNESS : 0;
    $rgb["blue"] = $rgb["blue"] > $DARKNESS ? $rgb["blue"]-$DARKNESS : 0;

    $c=imagecolorallocate($img,$rgb["red"],$rgb["green"],$rgb["blue"]);
    imagesetpixel($img, $x, $y, $c);
  }

$white=imagecolorallocate($img,255,255,255);
imagestring($img,2,530-100,18,"www.codenet.ru",$white);
imagestring($img,2,530-100,28,"(c) 2005",$white);

imagepng($img);
imagedestroy($img);
```

Результат работы этой программы выглядит следующим образом:

Практическая работа №42
Проектирование базы данных и обеспечение прав доступа к ней.

Постараемся показать, как получить доступ к базе данных из Web, используя PHP. На примерах будет также показано, как в простейшем варианте выбирать из базы и вносить в нее данные и как фильтровать потенциально опасную входную информацию. Как получить доступ к БД и прочитать из нее данные, покажет следующий простой пример формы поиска материалов. Форма проще некуда, и код между тегами <body> </body> приведен ниже:

```
<body> <h1> Denver's Auto Ware Search</h1>
<from action="result.php" method="post">
Выберите тип поиска: <br>
<select name="searchtype">
<option value="matcode"> по коду
<option value="matname"> по названию
</select> <br>
Введите строку поиска:<br>
<input name="searchterm" type=text> <br>
<input type=submit value="Поиск">
</form> </body>
```

В процессе работы пользователь заходит на страницу поиска материала, выбирает тип поиска и вводит искомую строку. После того, как пользователь нажимает на кнопку ПОИСК, вызывается сценарий result.php, исходный код которого приведен ниже с комментариями:

```
<html>
<head>
<title> Denver's Auto Ware Search Results</title>
</head>
<body>
<h1> Denver's Auto Ware Search</h1>
<? trim($searchterm);
```

Поясним, какие операции выполняет этот сценарий. Сначала необходимо убрать все лишние пробелы по краям слова, которые мог случайно или преднамеренно набрать пользователь. Функция trim() обрезает пробелы вначале и в конце строки. На следующем этапе нам необходимо убедиться, что пользователь указал тип поиска и строку поиска. Эта проверка должна проводиться уже после того, как лишние пробелы удалены из параметров поиска. В случае если не выбран тип поиска или не введена строка поиска, сценарий выдаст сообщение об ошибке и завершит свою работу:

```
if (!searchtype || !searchterm)
{echo "Вы не заполнили все поля для поиска. Вернитесь обратно и заполните все поля.";
exit}
```

Входные данные, как известно, проверять необходимо всегда. Вводимая информация не должна содержать управляющих символов. Поэтому подвергаем переменные $searchtype и $searchterm обработке функцией addslashes(). К данным, извлекаемым из базы данных в этом случае также необходимо применить функцию stripslashes().

```
$searchtype = addslashes($searchtype);
$searchterm = addslashes($searchterm);
```

Подключиться к серверу MySQL из PHP-сценария несложно. Достаточно вызвать функцию mysql_pconnect(), как показано в следующем примере:

```
@ $db = mysql_pconnect("localhost","Denver","den001");
```

Также можно использовать функцию mysql_connect(): она тоже возвращает идентификатор соединения. Разница состоит только в том, что функция mysql_pconnect() создает постоянное подключение, которое "живет" все время работы сценария.
В этих функциях требуется указать имя узла, на котором помещен сервер MySQL, опционально, через двоеточие можно указать порт, по которому следует подключаться, имя пользователя и пароль, чтобы войти на сервер. Все эти параметры являются необязательными, и если какой-либо не указать, функция воспользуется значением по умолчанию. В случае успеха функция возвращает идентификатор сеанса связи с базой данных или значение false в случае неудачи. Следующий далее код проверяет, успешно ли прошло соединение с базой данных, и показывает предупреждение, если соединение установить не удалось.

```
if (!db)
{echo "Ошибка: не могу соединиться с базой данных. Попробуйте еще раз позднее.";
exit}
```

Затем выбираем базу данных, с которой будем работать. Эта функция аналогична консольному вводу "use auto;". В результате будет использоваться именно выбранная база данных при формировании запросов к таблицам.

```
mysql_select_db("auto");
```

Далее настраиваем текст запроса и помещаем его в переменную $query. После этого вызываем функцию mysql_query(), которая и осуществляет запрос. Результат запроса помещаем в переменную $result для дальнейшего использования. Не забывайте, что запрос, передаваемый в MySQL, не требует точки с запятой в конце, в отличие от запроса, который вводится в среде монитора MySQL.

```
$query = "select * from materials where ".searchtype." like '%"$searchterm"%'";
$result = mysql_query($query);
```

Разнообразие функций дает нам возможность получить результат различными способами. Идентификатор результата — это ключ доступа к строкам, возвращенным запросом, которых может быть ноль, одна или несколько. Применяемая ниже функция mysql_num_rows() сообщает количество строк, которые возвращает запрос.

```
$num_results = mysql_num_rows ($result);
echo "<p> Найдено материалов: ".$num_results."</p> ";
```

Это полезно знать, если планируется обрабатывать или отображать результаты. Зная их количество, можно организовать цикл по строкам результата запроса.

```
for ($I = 0, $I < $num_results, $I++)
{$row = mysql_fetch_array ($result);
```

В каждой итерации цикла происходит вызов функции mysql_fetch_array(). Эта функция берет каждую строку из списка результата и возвращает ее в виде ассоциативного массива с ключом как именем атрибута. Имея $row в массиве, можно пройти каждое поле и отобразить его должным образом.

```
echo "<p> <strong> ".($I + 1).".Наименование: ";
echo htmlspecialchars( stripslashes($row["matname"]));
echo "</strong> <br> Код материала: ";
echo htmlspecialchars( stripslashes($row["matcode"]));
echo "</strong> <br> Спецификация: ";
echo htmlspecialchars( stripslashes($row["matspec"]));
echo "<br> Цена: ";
echo htmlspecialchars( stripslashes($row["cost"]));
echo "</p> ";} ?>
</body>
</html>
```

Существует несколько вариантов получения результата. Вместо ассоциативного массива можно воспользоваться нумерованным массивом, воспользовавшись функцией mysql_fetch_row(). Тогда значения атрибутов будут храниться в каждом порядковом значении $row[0], $row[1] и так далее. С помощью функции mysql_fetch_object() можно выбрать строку внутрь объекта и получать доступ к атрибутам как к свойствам объекта $row-> matcode, $row-> matname и так далее.
Каждый из этих вариантов подразумевает выборку строки за раз. Другой вариант — получить доступ, используя mysql_result(). Для этого потребуется указать сроку (от нуля до количества строк минус один) и название поля:

```
%row = mysql_result(%result, $I, "matcode");
```

Однако не стоит смешивать при работе mysql_result() с другими функциями выборки. Строчно-ориентированные функции выборки намного более удобны и эффективны, нежели mysql_result(), так что лучше использовать одну из них.
В случае, если есть необходимость закрытия непостоянного соединения, применяется

функция mysql_close(database_connection), однако применение ее спорно, так как с завершением выполнения сценария соединение закрывается автоматически. Однако рассмотренный выше поиск представляет только одну функциональную единицу приложения. Кроме организации поиска по базе, должно существовать еще как минимум и добавление. Не стоит забывать и об изменении данных, и об их удалении. Эти операции не сильно отличаются друг от друга — только структурой запроса. Так что при их проведении не должно возникнуть трудностей, синтаксис операторов insert, update и delete довольно стандартен. Рассмотрим их все на примерах.

Внесение новой информации очень похоже на получение существующей. Нужно пройти те же шаги: установить соединение, отправить запрос и проверить результаты. Только в данном случае вместо select будет использоваться insert.
Для успешной работы все, что вам нужно, — это создать HTML-страницу с формой, свойство action которой указывает на файл insert_mat.php с методом post. На форме должны присутствовать все поля, которые необходимо будет заполнить пользователю для внесения нового материала: код материала, его название, спецификацию и цену. Результаты заполнения формы передаются в insert_mat.php, а сценарий, занимающийся деталями, выполняет аутентификацию и пытается записать данные в базу данных. Структура его такая же, как и у сценария поиска. Для начала производим проверку полей на непустые значения. Если проверка показала, что не все поля заполнены, показываем пользователю сообщение об ошибке и прекращаем выполнение сценария. После этого преобразовываем значения полей ввода функциями addslashes() и doubleval(). Последняя функция применяется для фильтрации всех неподходящих символов в числовом поле. Соединение с базой данных происходит по вызову функции mysql_pconnect(), и, если оно было успешным, формируется строка запроса вставки новой записи в БД. В противоположном случае пользователю выдается сообщение об ошибке. После того, как запрос выполнен, можно проверить изменения в базе данных путем вызова функции mysql_affected_rows(), которая возвращает количество строк, которые были подвергнуты изменениям. Необходимо заметить, что, когда используется оператор select, применяется функция mysql_num_rows(), возвращающая количество строк в результате запроса. В случае работы с операторами insert, delete и update надо вызывать функцию mysql_affected_rows().

Теперь поговорим о внесении данных. Эта процедура очень похожа на процесс получения существующих данных. Нужно пройти те же шаги: установить соединение, отправить запрос и проверить результаты. Только в данном случае вместо Select будет использоваться Insert. Хоть все, вроде, и просто, но на пример никогда взглянуть не помешает. Исходя из количества полей для материалов на складе, мы создадим обычную HTML-форму для добавления новых материалов на склад, то есть в базу данных. Код этой страницы приведен ниже:

```
<html> <head> <title> Denver's Auto Ware</title> </head>
<body>
<h1> Denver's Auto Ware — Insert Material</1>
<form action="insert_mat.php" method="post">
<table border=0>
<tr> <td> Код материала</td>
<td> <input type=text name=mat_code> <br> </td> </tr>
<tr> <td> Название</td>
<td> <input type=text name=mat_name> <br> </td> </tr>
<tr> <td> Применение</td>
```

<td> <input type=text name=mat_spec>
 </td> </tr>
<tr> <td> Цена за ед.</td>
<td> <input type=text name=mat_cost>
 </td> </tr>
<tr> <td colspan = 2 > <input type=submit value = Добавить> </td> </tr>
</table> </form>
</body> </html>

Результаты заполнения этой формы передаются в сценарий insert_mat.php, а сценарий, занимающийся вставкой материала, выполняет определенного вида аутентификацию и пытается записать данные в базу.
Код сценария без HTML представлен ниже:

```
<?
If (!$mat_code || !$mat_name || !$mat_spec || !$mat_cost)
{echo "вы не заполнили все требуемые поля! Вернитесь назад и попробуйте снова.";
exit;}
```

В этой части мы снова проверяем введенные пользователем данные. По нашим условиям, он обязательно должен заполнить все поля без исключения, иначе материал в базу добавлен не будет. Идем по сценарию далее:

```
$mat_code = addslashes($mat_code);
$mat_name = addslashes($mat_name);
$mat_spec = addslashes($mat_spec);
$mat_cost = doubleval($mat_cost);
```

Следующие четыре строки делают переданные нам значения безопасными для скрипта и базы данных. Очевидно, что цены хранятся в базе данных в виде числа с плавающей запятой, символы наклонной черты они содержать не должны. Поэтому еще раз повторюсь: для чисел, в отличие от строковых значений, это достигается использованием функции doubleval применительно к числовому значению. Эта функция отфильтровывает все неподходящие символы в числовом поле. Она же, функция, позаботится и о всех символах валюты, которые пользователь может ввести.
Далее мы снова соединяемся с базой данных функцией mysql_pconnect():

```
@ $db = mysql_pconnect("localhost","Denver","den001");
if (!$db)
{echo "ОШИБКА: не могу соединиться с базой данных!";
exit;}

mysql_select_db("auto");
$query = insert into materials values ('".$mat_code."', '".$mat_name."', '".$mat_spec."',
'".$mat_cost."');
$result = mysql_query($query);
if ($result)
echo mysql_affected_rows()." Материалов добавлено в базу.";
?>
```

Одно существенное различие между операторами Select и Insert заключается в использовании mysql_affected_rows(). В предыдущем сценарии функция mysql_num_rows() применялась для определения количества строк, которые будет возвращать запрос Select. При написании запросов, которые изменяют базу данных, таких, как Insert, Delete, Update, следует использовать mysql_affected_rows(), которая возвращает количество строк в базе,

подвергнутых изменениям.
Итак, как накапливать данные в памяти и как с ними работать, уже известно. А как освободить память от них, если такая необходимость возникнет? Для освобождения ресурсов, а такая ситуация может действительно понадобиться при выполнении сценария, пригодится функция mysql_free_result(int result), которая освобождает память, занимаемую результатом. Очевидно, что все действия с данными должны производиться до вызова этой функции.

Из сценария можно не только работать с базой данных, но и создавать и удалять их. Для этого существуют функции mysql_create_db() и mysql_drop_db(). Обе функции используют имя базы данных и соединение. Если соединения нет, будет использоваться последнее открытое. В случае успешного выполнения функций они возвращают true, а в случае неудачи — false.
Для ужесточения мер безопасности не стоит забывать, что запускать сервер MySQL — mysqld от имени привилегированного пользователя не рекомендуется. Используя полученные права, пользователь MySQL может записывать и читать файлы в любом месте системы.

Практическая работа №43

Взаимодействие PHP и XML. Установка расширения DOM XML. Обработка элементов XML документа с помощью функций PHP. Использование XML-базы данных в качестве альтернативы реляционной СУБД.

Взаимодействие PHP и XML

Расширения SAX и DOM XML

Для работы с *XML* -документами можно использовать язык PHP. В **PHP** для этого существует два модуля, реализующие два разных стандарта обработки *XML* -данных: *SAX (Simple API for XML)* и *DOM (Document Object Model).*

Стандарт *SAX* (http://www.saxproject.org) не является стандартом W3C и описывает метод обработки *XML* -документов для получения из них данных. То есть этот метод обработки *XML* -документов позволит только прочитать данные из *XML* -документа, и не более того. Создавать и изменять *XML* -документы с его помощью невозможно.

SAX основан на так называемом *событийном программировании*. Его особенность заключается в том, что вы предоставляете парсеру *XML*набор собственных функций, которые будут заниматься обработкой различных типов *XML* -данных (*элементов* (тегов), текста и т.п.), а*нарсер* затем будет сам вызывать ваши функции в процессе обработки *XML* -документа, передавая им найденные данные. Функции будут вызываться в той же последовательности, в которой соответствующие данные располагаются в *XML* -документе.

Другим стандартом для обработки *XML* -данных является *DOM* – стандарт **W3C**, спецификацию которого можно найти на сайте консорциума (http://www.w3c.org/DOM). В отличие от *SAX*, этот метод позволяет производить любые операции с *XML* -данными в достаточно удобной форме – представляя *XML* -документ как дерево объектов.

Модуль, реализующий этот стандарт, называется *DOM XML*. Он не входит в основной набор модулей PHP, но может быть установлен как расширение. API этого модуля старается как можно более точно следовать стандарту *DOM* level 2. Кроме того, существует множество дополнительных функций. Эти функции включены для совместимости с

предыдущими версиями расширения, и использовать их в новых скриптах не рекомендуется. Кроме того, у расширения *DOM XML* есть проблемы с русской кодировкой. *Парсер* обрабатывает текст только в кодировке UTF-8, поэтому текст нужно каждый раз перекодировать с помощью функции iconv. Отсюда и необходимость установки расширения iconv вместе с *DOM XML*.

Модуль *DOM XML* является мощным и удобным в использовании средством обработки *XML* -документов. В данной лекции мы будем рассматривать именно его.

Установка расширения DOM XML

Для того чтобы установить расширение *DOM XML*, нужно сделать следующее.

- В *файле настроек* PHP (*php.ini*) раскомментировать строку, касающуюся этого расширения (extension=php_domxml.dll для Windows, либо extension=php_domxml.so для Linux-платформ).
- Скопировать файл расширения (php_domxml.dll или php_domxml.so) в папку, где находятся расширения (*extension_dir*).
- Подключить расширение iconv так же, как в пунктах выше (иногда это расширение устанавливается автоматически вместе cdomxml).
- Скопировать дополнительные библиотеки в системную папку system (Windows 98) или system32 (WindowsNT/2000/XP). В первую очередь это библиотеки libxml2 и iconv, затем libxslt, libexslt и zlib.
- Перезапустить сервер.

Следует проверить, правильно ли установлена переменная *extension_dir* в файле настройки *php.ini*. Если она не указывает на директорию, где находятся библиотеки расширений PHP, то ни одно из расширений подключить не удастся.

Чтобы проверить, установилось ли расширение, можно создать простейший скрипт, который будет выводить все настройки PHP-интерпретатора (это делает функция *phpinfo()*). Другой вариант – попробовать использовать какую-нибудь функцию из данного расширения. Например, можно попробовать получить версию используемой библиотеки libxml с помощью функции *domxml_version()* . Но этот способ не очень хорош, поскольку расширение экспериментальное (это значит, что некоторые функции в каких-то определенных условиях могут и не работать), да и функции еще надо изучить, прежде чем их использовать.

```
<?
// выводит информацию о настройках PHP
phpinfo();
// отображает используемую версию
// библиотеки libxml
echo domxml_version();
?>
```

Пример 14.1. Проверка, установлено ли расширение DOM XML

[+]
Записаться
|
Вам нравится? Нравится 212 студентам
| Поделиться |
Поддержать курс
| Скачать электронную книгу

Поделиться
Лекция 14:
Взаимодействие PHP и XML
A
|
версия для печати
< Лекция 13 || **Лекция 14**: 12345 || Лекция 15 >

Взаимодействие PHP и XML посредством DOM XML

Что происходит, если взаимодействие *PHP* и *XML* осуществляется с помощью *объектной модели* стандарта *DOM*? *Модуль DOM XML*определяет в *PHP* несколько классов, таких как *DomNode*, *DomDocument*, *DomElement*, *DomText* и *DomAttribute*, большинство из которых идут из ядра стандарта *DOM*. Почти для всех классов (в частности, для перечисленных выше) *класс DomNode* является *родительским*, поэтому его свойства и методы наследуются всеми остальными классами.

Если рассмотреть произвольный *XML* -документ, то классу *DomDocument* будет соответствовать сам этот документ, классу *DomElement* – каждый *XML* -тег, классу *DomAttribute* – атрибуты тегов, а классу *DomText* – содержание XML- *элементов*. В то же время классу *DomNode*будет соответствовать каждый из перечисленных *элементов XML* -документа.

Рассмотрим коллекцию, содержащую описания персон. Если каждую из них мы описываем с помощью таких характеристик, как фамилия, имя, дата рождения и электронный *адрес*, то *структура* коллекции " Личности ", где хранится *информация* обо всех известных нам персонах, может быть представлена следующим образом.

```
<?xml version="1.0"?>
<collection>
   <person id="10">
      <name>
         <first>Nick</first>
         <last>Petrov</last>
      </name>
      <birth>
         <day>23</day>
         <month>12</month>
         <year>89</year>
      </birth>
      <email> nick@ngs.ru
      </email>
```

```
  </person>
  <person id="20">
    <name>
      <first>Bob</first>
      <last>Ivanov</last>
    </name>
    <birth>
      <day>03</day>
      <month>05</month>
      <year>90</year>
    </birth>
    <email> bob@ngs.ru
    </email>
  </person>
</collection>
```

Пример 14.2. Коллекция Личности в виде XML-файла (persons.xml)

В дальнейшем, приводя примеры, мы будем использовать этот *файл*.

Нам необходимо научиться читать, добавлять, изменять и искать информацию, находящуюся в *XML* -файлах.

Перевод данных XML-файла в объекты и классы PHP

Первое, что нужно сделать, если мы хотим работать с *XML* -данными в PHP при помощи расширения *DOM XML*, это перевести имеющиеся данные в объекты и классы *DOM*. Это можно сделать несколькими способами.

1. С помощью функции *domxml_open_mem* .

   Синтаксис:

   ```
   object domxml_open_mem (string str)
   ```

   В качестве параметра эта функция принимает строку str, содержащую *XML* -документ. Результатом ее работы является объект класса, называемого *DOMDocument*.
2. С помощью функции *domxml_open_file* .

   Синтаксис:

   ```
   object domxml_open_file (string filename)
   ```

   Эта функция обрабатывает *XML* -файл, имя которого задается параметром filename, и переводит его в объект класса *DOMDocument*. Доступ к файлу производится только на чтение.

Такие функции, как *domxml_open_mem()* и *domxml_open_file()* , как правило, нужно вызывать перед вызовом любых других функций, связанных с расширением *DOM*.

Эти функции преобразуют *XML* -файл в дерево объектов. К таким объектам можно обращаться с помощью различных методов. В частности, для выделения *корневого элемента* используется метод *DomDocument->document_element()* .

Еще существует функция *domxml_new_doc(string version)* , которая создает новый пустой *XML* -документ. Ее параметром является номер версии создаваемого документа. Но ее мы касаться не будем, а будем считать, что *XML* -файл уже создан.

```
<?
//считываем файл "persons.xml" в строку
$xmlstr = join('',file('persons.xml'));
// переводим строку с xml-файлом
// в дерево объектов. Если операция
// прошла неудачно, то выводим
// ошибку и прекращаем работу.
if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Ошибка при разборе документа\n";
  exit;
}
// можно посмотреть, как выглядит
// этот объект
print_r($dom);
echo "<hr>";
// выделяем корневой элемент
// дерева объектов.
// В нашем случае это будет
// элемент <collection>
$root = $dom->document_element();
print_r($root);
echo "<hr>";
?>
```

Пример 14.3. Перевод XML-файла в дерево объектов PHP и выделение корневого элемента

Итак, каждому *элементу XML* -файла мы поставили в соответствие какой-то объект. Теперь нужно научиться перемещаться по дереву объектов и обращаться с этими объектами: получать и изменять их значения, находить их *потомков* и предков, удалять объекты.

Обход дерева объектов

Для получения значения текущего узла (вне зависимости от его типа) используют метод *DomNode->node_value()* или *DomNode->get_content()* для получения содержимого узла.

Для получения значения атрибута используется метод *DomElement->get_attribute (attr_name)* . А метод *DomNode->child_nodes()* возвращает массив *потомков* данного узла.

Для того чтобы сделать обход дерева объектов, полезно еще уметь различать объекты по типам, т.е. определять, является ли узел*элементом* (тегом), текстом, атрибутом и т.п. Для этого используются специальные константы. *XML_ELEMENT_NODE* определяет, является

ли узел *элементом*, *XML_ATTRIBUTE_NODE* определяет, является ли узел атрибутом, и *XML_TEXT_NODE* определяет, является ли узел куском текста. Эти константы имеют целочисленные значения 1, 2 и 3 соответственно. Использование этих констант полезно, поскольку переводы строки, применяемые для *удобочитаемости XML* -файлов, тоже становятся узлами.

```
<?
// сначала делаем то же,
// что и в предыдущем примере
$xmlstr = join('',file('persons.xml'));
if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Ошибка при разборе документа\n";
  exit;
}
$root = $dom->document_element();
// Получаем массив потомков
// родительского узла
// (в нашем случае это массив <person>)
$nodes = $root->child_nodes();
print_r($nodes);
echo "<hr>";
// Начинаем обработку каждого
// узла в массиве
foreach($nodes as $node){
    // Если текущий узел – один
    // из узлов <person>, то
    // продолжаем ее обработку,
    // чтобы получить информацию
    // об этой личности
    if ($node->tagname=='person'){
    // Создаем массив, куда
    // будем собирать информацию
    // о рассматриваемой личности
    $currentPers = array();
    // Получаем id личности,
    // который хранится в атрибуте 'id'
    $currentPers['id'] =
         $node->get_attribute('id');
    // Получаем массив потомков
    // <person>. Это вся
    // информация о личности
    // (<name>,<birth> и т.д.)
    $persons_info =
              $node->child_nodes();
    // Перебираем все дочерние
    // узлы $node
    foreach ($persons_info as $info){
    // проверяем, является ли узел
    // элементом (xml-тегом)
```

```
   if ($info->type==
            XML_ELEMENT_NODE) {
      // тогда метод tagname
      // возвратит имя этого
      // элемента (тега), а метод
      // get_content() –
      // его содержимое
      $currentPers[$info->tagname] =
            $info->get_content();
      }
   }
   // выводим на экран полученные
   // массивы
   print_r ($currentPers);

   echo "<br>";
   }
}
?>
```

Пример 14.4. Обход дерева XML

Итак, мы научились обходить дерево *XML*. Теперь можно попытаться что-нибудь найти в *XML* -файле. Правда, делать это не совсем удобно опять же из-за переносов строк, которые мы использовали при написании *XML* -файла. Пусть наш *XML* -файл записан в строку, а точнее, в нем есть следующая строка:

```
...
<person id="20"><name><first>Иван</first><last>Иванов</last></name>
...
```

Тогда в наш предыдущий пример вставим (после вывода на экран полученных массивов) строчку для поиска электронного адреса Ивана Иванова.

```
...
   $str = $currentPers["email"];
   if ($currentPers["name"] ==
            "Иван Иванов" )
      echo "Здравствуйте, Иван! " .
      "Ваш e-mail $str";
...
```

Добавление новых элементов в XML-документ

Далее разберем задачу, как можно добавить в нашу базу данных новую личность средствами php.

Сначала нужно скопировать описание личности (считаем, что все личности описываются с помощью стандартного набора характеристик, как в файле persons.xml). Это делается с помощью метода *DomNode->clone_node()* . Таким образом, мы клонируем *элемент*<person> и все его внутренние *элементы* (содержание тегов не копируется).

Потом можно установить какие-нибудь значения для *элементов* описания личности. Например, задать имя человека, дату его рождения и т.п. В конце нужно записать полученное описание личности в качестве *потомка корневого элемента* в дерево *DOM* с помощью метода*DomNode->append_child(new_node)* , где в качестве параметра передается созданный объект (новый узел).

В PHP до версии 4.3 перед добавлением *потомка* к узлу с помощью данной функции этот *потомок* сначала копировался. Таким образом, новый узел являлся новой копией, которая могла изменяться без изменения узла, переданного как параметр в эту функцию. В более поздних версиях PHP новый узел удаляется из существующего контекста, если он уже есть в дереве. Такое поведение соответствует спецификациям W3C.

Для *удаления узла* можно воспользоваться методом, применив его к узлу, который требуется удалить, т.е. *DomNode->unlink_node()* .

```
// Для того чтобы добавить описание
// новой личности, нужно знать,
// как описывается каждая личность.
// Выбираем элемент person,
// который содержит описание личности
$elements = $dom->get_elements_by_tagname("person");
$element = $elements[0];
//вычисляем родителя и потомков
$parent = $element->parent_node();
$children = $element->child_nodes();
// клонируем элемент person
$person = $element->clone_node();
// устанавливаем новой
// личности идентификатор
$attr = $person->set_attribute("id", "30");
// если у личности были потомки,
// то их тоже надо клонировать
foreach ($children as $child){
  //клонируем ребенка
  $node = $child->clone_node();
  //получаем массив внуков
  $grand_children = $child->child_nodes();
  // если ребенок имеет потомков,
  //т.е. массив внуков не пуст, то
  if (count($grand_children)<>1){
      //клонируем каждого внука
      //и присоединяем к уже
      //клонированному ребенку
```

```
    foreach($grand_children as $grand_child){
     $lastnode = $grand_child->clone_node();
       //записываем в нужные теги
       //подходящие значения
      if ($grand_child->tagname=="first")
       $cont = $lastnode->set_content("Nina");
      if ($grand_child->tagname=="last")
       $cont = $lastnode->set_content("Saveljeva");
      if ($grand_child->tagname=="day")
       $cont = $lastnode->set_content("7");
      if ($grand_child->tagname=="month")
       $cont = $lastnode->set_content("06");
      if ($grand_child->tagname=="year")
       $cont = $lastnode->set_content("1981");
      $newlastnode = $node->append_child($lastnode);
        }
    }
    if ($child->tagname=="email") {
       $cont = $node->set_content("help@intuit.ru");
    }
    $newnode2 = $person->append_child($node);
}
$newnode = $parent->append_child($person);
//dump_mem создает XML-документ из dom
//представления
echo "<PRE>";
$xmlfile = $dom->dump_mem(true);
// посмотрим в браузере,
// что получилось
echo htmlentities($xmlfile);
echo "</PRE>";
// запишем полученный XML-файл
// в файл "test.xml"
$h = fopen("test.xml","a");
if (!fwrite($h, $xmlfile)) {
       print "Cannot write " . "to file ($filename)";
       exit;
    }
}
```

Пример 14.5. Добавление описания новой личности в каталог

Заключение

Итак, мы изучили ряд функций, позволяющих манипулировать данными, хранящимися в *XML* -формате. Это, конечно же, далеко не полный перечень существующих функций. В версии *PHP5* он значительно усовершенствован и в большей степени соответствует стандарту *DOM*. Тем не менее *знание* приведенных здесь основных функций может оказаться полезным при решении конкретных прикладных задач.

Практическая работа №44
Использование шаблонов в PHP. Понятие шаблона и его использование в языке программирования PHP. Классы шаблонов FastTemplate и Smarty.

Использование шаблонов

Прежде чем использовать шаблоны, подумайте, действительно ли они вам так нужны? В данный момент существует огромное количество коммерческих вариантов шаблонов. Все они работают по одному принципу (значение, замена), но имеют огромное количество наворотов, таких как автоматическое изменения регистра переменных, поиск по регулярным выражениям и т.д., все это конечно хорошо и легко реализуемо. Когда я решил посмотреть "коммерческий" шаблон, я ужаснулся, один его класс весил 398 КБ. Это нормально? Также в сети можно найти множество бесплатных вариантов шаблонов (классы шаблонов в PHPBB, IPB...), но все они много весят и работают не слишком быстро. Я предлагаю вам простой каркас "шаблонов" на PHP, с его помощью можно сделать свой классный шаблонизатор, со всеми необходимыми вам функциями.

За и против

Приведу вам жизненный пример, не так давно я занимался разработкой программы для одного человека, заранее было обговорено, что я пишу программу, а дизайн это его дело. Через некоторое время, мой заказчик пишет мне, что дизайн для моей программы сделать невозможно. Конечно, человек ничего не знающий в web-программировании будет испытывать огромные затруднения, при построении дизайна в PHP-программе. Главная задача "шаблонов" – это облегчить жизнь дизайнеру. Безусловно, главным плюсом использования шаблонов можно считать то, что дизайнер без помощи программиста сможет изменять свой web-проект. Также мне нравится само разделение – программа и дизайн.

Я не использую шаблоны в своих личных проектах, т.к. они дают дополнительную "нагрузку". Шаблоны это хорошо, но использовать их надо только если пишешь какой, то публичный проект или выполняешь работу на заказ.

Реализация шаблонов на PHP

И так приступим. Всего у нас будет 2 ключевых файла.

1) file2compile.tpl – файл который мы будем парсить
2) template.php – главный файл содержащий класс шаблонов

Листинг файла file2compile.tpl:

```
<html>
<head>{TITLE}</head>
<body bgcolor={BGCOLOR}>

{SOMETPLTAGS}

</body>
```

```
</html>
```

Листинг файла template.php:

```
<?php
class parse_class
  {
  var $vars   = array();
  var $template;

  function get_tpl($tpl_name)
   {
   if(empty($tpl_name) || !file_exists($tpl_name))
    {
    return false;
    }
   else
    {
    $this->template = file_get_contents($tpl_name);
    }
   }
  function set_tpl($key,$var)
   {
   $this->vars[$key] = $var;
   }
  function tpl_parse()
   {
   foreach($this->vars as $find => $replace)
      {
      $this->template = str_replace($find, $replace, $this->template);
      }
   }
  }
$parse = new parse_class;
?>
```

Теперь я подробно опишу содержание этих двух файлов.

Файл: file2compile.tpl

Тут приведен обычный HTML код. В данном файле можно найти переменные вида {TITLE}. Это как раз именно те переменные которые мы будем заменять на нужное нам значение.

Файл: template.php

Мы имеем PHP класс, разделенный на 3 функции. В самом начале файла мы объявляем классовые переменные.

$vars – массив со значениями (переменная, замена).
$template – файл который мы будем парсить.

Массив в php - это упорядоченное отображение, которое устанавливает соответствие между значением и ключом.

Теперь перейдем к описанию функций.

Функция: get_tpl

В качестве аргумента функция принимает имя файла. В теле функции мы проверяем задан ли аргумент и существует ли файл. Если аргумент не задан и файл не существует мы возвращаем значение FALSE. В противном случае мы заполняем классовую переменную(template) содержанием файла.

Функция set_tpl

Функция принимает 2 значения, это переменная (напр. {TITLE)} и значение на которое мы будем ее заменять.

Функция tpl_parse

Функция не принимает никаких значений. В теле функции мы считывает массив $vars и производим замену установленных переменных на заданные значения.

Использование класса.

Для вывода на экран используйте следующие команды:

```
require('template.php'); // Подключаем файл с классом
$parse->get_tpl('template.tpl'); // Файл который мы будем парсить
$parse->set_tpl('{TITLE}','Супер сайт'); // Установка переменной {TITLE}
$parse->set_tpl('{BGCOLOR}','#F2F2F2'); // Установка переменной { BGCOLOR }
$parse->set_tpl('{SOMETPLTAGS}','Текст красным'); //Установка переменной {SOMETPLTAGS}
$parse->tpl_parse(); // Парсим
print $parse->template; // Выводим нашу страничку
```

Практическая работа №45
Работа с Cookies в PHP. Авторизация доступа с помощью сессий. Конфигурация PHP для работы с сессиями. Отслеживание сеанса в PHP.

Откуда возник термин "cookie" никто достоверно не знает, хотя считается, что во времена зарождения Unix-систем где-то использовалось словосочетание Magic Cookies. Имелись в виду "квитанции" (token, ticket), которыми обменивались программы.

Cookie является решением одной из наследственных проблем HTTP протокола (HyperText Transfer Protocol). Эта проблема заключается в непостоянстве соединения между клиентом

и сервером, как при FTP или Telnet сессии, т.е. для каждого документа (или файла) при передаче по HTTP протоколу посылается отдельный запрос. Включение cookie в HTTP протокол дало частичное решение этой проблемы. Иначе говоря, транзакция завершается после того, как браузер сделал запрос, а сервер выдал соответствующий ответ. Сразу после этого сервер "забывает" о пользователе и каждый следующий запрос того же пользователя считает новым пользователем.

Используя cookie, можно эмулировать сессию по HTTP протоколу. Коротко принцип эмуляции сессии таков: на первом запросе выдается соотвествующее значение cookie, а при каждом последующем запросе это значение читается из переменной окружения HTTP_COOKIE и соответствующим образом обрабатывается.

Простой пример: есть форма, где пользователю предлагается указать свое имя, из нее вызывается скрипт, прописывающий значение cookie в браузер пользователя. При каждом последующем заходе на основе анализа значения cookie из браузера пользователя на странице появляется либо именное приветствие (если есть установленное значение cookie), либо первоначальная форма с запросом имени пользователя (если значение cookie не установлено).

Итак,приступим к практике:

1.Задание cookie с помощью Php

Для задания этой функции в языке php есть оператор: setcookie() . Самое приятное,сто функция setcookie() воспринимает до шести аргументов, в зависимости от того, как вы собираетесь управлять значениями cookie и кто будет считывать ее значения.

Простейший способ установить cookie таков:

```
setcookie('name', 'bret');
```

Затем, для каждой последующей страницы на Вашем сайте, просматриваемой в течение данной сессии (пока пользователь не покинет сайт) переменная $name будет иметь значение 'bret' и его можно легко прочитать средствами PHP. Этот тип cookie известен как cookie-сессия, поскольку значение сохраняется в течение пользовательской сессии.

Если Вы хотите, чтобы значение cookie запоминалось браузером после того, как пользователь закончит сессию, Вы должны передать функции setcookie() третий параметр - дату истечения срока действия cookie. Поскольку PHP сформировался в основном в среде Unix, Вы должны представить время истечения срока действия cookie как число секунд, прошедших с 1 января 1970 г. Если Вы имеете опыт программирования для Unix, это не покажется Вам удивительным. Но, если Вы программировали только в среде Windows или Macintosh, Вы, может быть, удивитесь, что за чокнутый народ эти Unix-оиды.

Но не бойтесь. PHP имеет очень удобную функцию, mktime() . Вы указываете ей в качестве параметров (в указанном порядке) час, минуту, секунду, месяц, день и год, задающие тот момент времени, который Вы хотите представить в воспринимаемом UNIX формате, и mktime() возвращает Вам число секунд, прошедших с 1 января 1970 г. до указанного момента времени. Например, если Вы хотите, чтобы срок действия cookie истек 1 января 2000 г., Вы записываете:

```
<?php
```

```
$y2k = mktime(0,0,0,1,1,2000);

setcookie('name', 'bret', $y2k);

?>
```

Если Вы хотите изменить значение cookie на новое, Вы можете просто переписать его (ее?) значение. Таким образом, даже если браузер уже посылал значение cookie серверу на одной из предыдущих страниц, вполне возможно сообщить серверу, что в действительности Вас зовут "jeff."

```
<?php

$y2k = mktime(0,0,0,1,1,2000);

setcookie('name', 'jeff', $y2k);

?>
```

Обратите внимание на то, что при этом не меняется значение переменной $name. Оно устанавливается при загрузке страницы. Если Вы хотите чтобы значение переменной изменялось синхронно с изменением значения cookie, Вы должны изменить код следующим образом:

```
<?php

$name = 'jeff';

$y2k = mktime(0,0,0,1,1,2000);

setcookie('name', $name, $y2k);

?>
```

Следующие два параметра функции setcookie() позволяют Вам задать путь и имя домена того, кто может прочитать значение Вашего cookie. По умолчанию только страницы, расположенные в том же каталоге или ниже в структуре подкаталогов того сервера, который установил cookie, могут прочитать его (ее??) значение. Это делается из соображений безопасности. Однако, если у Вашего сервера два доменных имени: "www.domain.com" и "other.domain.com", и Ваш экаунт позволяет Вам обслуживать страницы из каталога ~/myhome, Вы должны вызывать функцию setcookie() следующим образом:

```
<?php

setcookie('name', 'jeff', $y2k, '~/myhome', '.domain.com');

?>
```

Последний параметр функции setcookie() , который мы никогда не использовали, требует, чтобы значение cookie передавалось только на те Web-сервера, которые испольуют безопасный протокол соединения, такой как SSL. Если Вам это нужно, то задайте для шестого параметра значение 1.

Удалить cookie тоже очень просто, достаточно передать функции setcookie() имя cookie и PHP сделает все остальное:

```
<?php setcookie('name'); ?>
```

В заключение нужно сделать еще одно замечание, касающееся использования cookie. В силу того, как организована обработка cookies в протоколе HTTP, необходимо установить значения всех cookie до вывода какого-либо текста. Если сделать наоборот, PHP выдаст Вам предупреждение и значение cookie не будет послано. Вот так правильно:

```
<?php
setcookie('name', 'jeff');
echo "Hello Everyone!";
?>
```

А так - нет:

```
<?php
echo "Hello Everyone!";
setcookie('name', 'jeff');
?>
```

2. Задание cookie с помощью JavaScript

Можно задавать значение cookie, используя язык JavaScript. Единственный недостаток этого способа заключается в том, что не все браузеры его поддерживают. Ниже приведены примеры функций JavaScript, написанные Алексеем Александровым для скрипта "Органайзер".

Пример. Функция установки значения cookie

```
// name - имя cookie
// value - значение cookie
// [expires] - дата окончания действия cookie (по умолчанию - до конца сессии)
// [path] - путь, для которого cookie действительно (по умолчанию - документ, в котором значение было установлено)
// [domain] - домен, для которого cookie действительно (по умолчанию - домен, в котором значение было установлено)
// [secure] - логическое значение, показывающее требуется ли защищенная передача значения cookie

function setCookie(name, value, expires, path, domain, secure) {
var curCookie = name + "=" + escape(value) +
((expires) ? "; expires=" + expires.toGMTString() : "") +
((path) ? "; path=" + path : "") +
((domain) ? "; domain=" + domain : "") +
```

((secure) ? "; secure" : "")

if (!caution || (name + "=" + escape(value)).length

Пример. Функция чтения значения cookie

Возвращает установленное значение или пустую строку, если cookie не существует.

// name - имя считываемого cookie

function getCookie(name) {

var prefix = name + "="

var cookieStartIndex = document.cookie.indexOf(prefix)

if (cookieStartIndex == -1)

return null

var cookieEndIndex = document.cookie.indexOf (";", cookieStartIndex + prefix.length)

if (cookieEndIndex == -1)

cookieEndIndex = document.cookie.length

return unescape(document.cookie.substring (cookieStartIndex + prefix.length, cookieEndIndex))

}

Пример. Функция удаления значения cookie

Принцип работы этой функции заключается в том, что cookie устанавливается с заведомо устаревшим параметром expires, в данном случае 1 января 1970 года.

// name - имя cookie

// [path] - путь, для которого cookie действительно

// [domain] - домен, для которого cookie действительно

function deleteCookie(name, path, domain) {

if (getCookie(name)) {

document.cookie = name + "=" +

((path) ? "; path=" + path : "") +

((domain) ? "; domain=" + domain : "") +

"; expires=Thu, 01-Jan-70 00:00:01 GMT"

}

3. Задание cookie с помощью Perl

Самый мощный и гибкий способ управления документами с использованием механизма cookie - с помощью CGI-скриптов. Задание значения cookie на Perl будет выглядеть следующим образом:

print "Content-type: text/htmln";

print "Set-Cookie: username=aaa13; expires=Friday, 31-Dec-99 23:59:59 GMT; path=/; domain=www.citforum.ru;nn";

Скрипт при выдаче результатов работы генерирует HTTP заголовок:

Content-type: text/html

Set-Cookie: "username=aaa13; expires=Friday, 31-Dec-99 23:59:59 GMT; path=/; domain=www.webscript.ru;"

Чтобы прочитать в скрипте ранее заданное значение cookie, используется переменная окружения HTTP_COOKIE.

$cookie = $ENV{'HTTP_COOKIE'};

Далее можно анализировать полученную строку и, в зависимости от считанных значений, выполнять соответствующие действия.

А теперь о грусном...

Ограничения:

Клиент (браузер) имеет следующие ограничения для cookies:

- всего может храниться до 300 значений cookies
- каждый cookie не может превышать 4Кбайт
- с одного сервера или домена может храниться до 20 значений cookie

Если ограничение 300 или 20 превышается, то удаляется первая по времени запись. При превышении лимита объема в 4Кбайт корректность значения cookie страдает - отрезается кусок записи (с начала этой записи) равный превышению объема.

В случае кэширования документов, например, proxy-сервером, поле Set-cookie HTTP заголовка никогда не кэшируется.

Если proxy-сервер принимает ответ, содержащий поле Set-cookie в заголовке, предполагается, что поле доходит до клиента вне зависимости от кода возврата 304 (Not Modified) или 200 (OK). Соответственно, если клиентский запрос содержит в заголовке Cookie, то он должен дойти до сервера, даже если жестко установлен параметр If-modified-since.

СПИСОК ЛИТЕРАТУРЫ

Основные источники:

1. Золотов С.Ю. Проектирование информационных систем [Электронный ресурс] : учебное пособие / С.Ю. Золотов. — Электрон.текстовые данные. — Томск: Томский государственный университет систем управления и радиоэлектроники, Эль Контент, 2013. — 88 с. — 978-5-4332-0083-8. — Режим доступа: http://www.iprbookshop.ru/13965.html

2. Дружинин Г.В. Эксплуатационное обслуживание информационных систем [Электронный ресурс] : учебник / Г.В. Дружинин, И.В. Сергеева. — Электрон.текстовые данные. — М. : Учебно-методический центр по образованию на железнодорожном транспорте, 2013. — 220 c. — 978-5-9994-0035-2. — Режим доступа: http://www.iprbookshop.ru/16268.html

3. Золотарёв О.В. Технология внедрения корпоративных информационных систем [Электронный ресурс] : методические указания к лабораторным работам / О.В. Золотарёв. — Электрон.текстовые данные. — М. : Российский новый университет, 2013. — 40 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/21325.html

4. Душин В.К. Теоретические основы информационных процессов и систем [Электронный ресурс] : учебник / В.К. Душин. — Электрон.текстовые данные. — М. : Дашков и К, 2014. — 348 c. — 978-5-394-01748-3. — Режим доступа: http://www.iprbookshop.ru/24764.html

5. Павлова Е.А. Технологии разработки современных информационных систем на платформе Microsoft.NET [Электронный ресурс] / Е.А. Павлова. — Электрон.текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 128 c. — 978-5-9963-0003-7. — Режим доступа: http://www.iprbookshop.ru/52196.html

6. Бурков А.В. Проектирование информационных систем в Microsoft SQL Server 2008 и VisualStudio 2008 [Электронный ресурс] / А.В. Бурков. — Электрон.текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 310 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/52166.html

7. Нестеров С.А. Анализ и управление рисками в информационных системах на базе операционных систем Microsoft [Электронный ресурс] / С.А. Нестеров. — Электрон.текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 250 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/52141.html

8. Федотов Е.А. Администрирование программных и информационных систем [Электронный ресурс] : учебное пособие / Е.А. Федотов. — Электрон.текстовые данные. — Белгород: Белгородский государственный технологический университет им. В.Г. Шухова, ЭБС АСВ, 2012. — 136 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/27280.html

9. Информационные системы и технологии в экономике и управлении. Проектирование информационных систем [Электронный ресурс] : учебное пособие / Е.В. Акимова [и др.]. — Электрон.текстовые данные. — Саратов: Вузовское образование, 2016. — 178 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/47671.html

10. Модели информационных систем [Электронный ресурс] : учебное пособие / В.П. Бубнов [и др.]. — Электрон.текстовые данные. — М. : Учебно-методический центр по

образованию на железнодорожном транспорте, 2015. — 188 c. — 978-5-89035-833-2. — Режим доступа: http://www.iprbookshop.ru/45279.html

11. Стасышин В.М. Проектирование информационных систем и баз данных [Электронный ресурс] : учебное пособие / В.М. Стасышин. — Электрон.текстовые данные. — Новосибирск: Новосибирский государственный технический университет, 2012. — 100 c. — 978-5-7782-2121-5. — Режим доступа: http://www.iprbookshop.ru/45001.html

12. Абденов А.Ж. Методика оценки риска для информационных систем на основе экспертных оценок [Электронный ресурс] : учебное пособие / А.Ж. Абденов, С.А. Белкин, Р.Н. Заркумова-Райхель. — Электрон.текстовые данные. — Новосибирск: Новосибирский государственный технический университет, 2014. — 71 c. — 978-5-7782-2588-6. — Режим доступа: http://www.iprbookshop.ru/44957.html

13. Терещенко П.В. Интерфейсы информационных систем [Электронный ресурс] : учебное пособие / П.В. Терещенко, В.А. Астапчук. — Электрон.текстовые данные. — Новосибирск: Новосибирский государственный технический университет, 2012. — 67 c. — 978-5-7782-2036-2. — Режим доступа: http://www.iprbookshop.ru/44931.html

Дополнительные источники:

1. Леонидова Г.Ф. Программно-техническое обеспечение автоматизированных библиотечно-информационных систем. Часть 2. Программное обеспечение автоматизированных библиотечно-информационных систем [Электронный ресурс] : учебное пособие для студентов специальности 071201 «Библиотечно-информационная деятельность» / Г.Ф. Леонидова. — Электрон.текстовые данные. — Кемерово: Кемеровский государственный институт культуры, 2012. — 264 c. — 978-5-8154-0221-8. — Режим доступа: http://www.iprbookshop.ru/22065.html

2. Ивницкий В.А. Моделирование информационных систем железнодорожного транспорта [Электронный ресурс] : учебное пособие / В.А. Ивницкий. — Электрон.текстовые данные. — М. : Учебно-методический центр по образованию на железнодорожном транспорте, 2015. — 276 c. — 978-5-89035-855-4. — Режим доступа: http://www.iprbookshop.ru/45280.html